1ST EDITION

TradeStation EasyLanguage for Algorithmic Trading

Discover real-world institutional applications of Equities, Futures, and Forex markets

DOMENICO D'ERRICO

TradeStation EasyLanguage for Algorithmic Trading

Copyright © 2024 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing or its dealers and distributors, will be held liable for any damages caused or alleged to have been caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

Group Product Manager: Kunal Sawant

Publishing Product Manager: Akash Sharma

Book Project Manager: Manisha Singh

Senior Editor: Nithya Sadanandan

Technical Editor: Vidhisha Patidar

Copy Editor: Safis Editing

Language Support Editor: Safis Editing

Proofreader: Nithya Sadanandan

Indexer: Pratik Shirodkar

Production Designer: Shankar Kalbhor

Senior DevRel Marketing Coordinator: Shrinidhi Manoharan

First published: September 2024

Production reference: 1100924

Published by Packt Publishing Ltd.

Grosvenor House

11 St Paul’s Square

Birmingham

B3 1RB, UK.

ISBN 978-1-83588-120-0

www.packtpub.com

My thanks go to everyone who, during my adventures in financial markets, has taught me—sometimes unintentionally—how things work and how algorithms are just a small part of a much more complex world where human relationships and psychological aspects play the true key role.

A special thanks to Antonio Zaffino, whose extensive trading experience in the markets has taught me that when navigating financial markets, it's better to make mistakes on my own than by following someone else's advice.

A special thanks to the team at The Gandalph Project and to Giovanni Trombetta, without whom the idea of AI applied to financial markets would still be a closed book to me.

- Domenico D'Errico

Contributors

About the author

Domenico D’Errico is a **Certified SIAT Technical Analyst** (**CSTA**). After having held various managerial roles within multinational companies, he has been working as a quant developer for algorithmic hedge funds for the past 15 years, being involved in several trading ventures.

He is an EasyLanguage specialist and a two-time winner of the 2011 TradeStation Developers contest. He published several articles in the Stocks & Commodities magazine.

He has developed a strong interest in AI research applied to the financial world, finding numerous similarities between some AI techniques and traditional technical trading.

Domenico is based in Matera, Italy. If you happen to be in the area or need anything, feel free to reach out to him at dderrico@tradingalgo.it .

About the reviewer

Giovanni Trombetta is an entrepreneur and electronic engineer specializing in AI algorithms applied to quantitative finance.

Since 2012, he has been the founder and head of R&D at the Gandalf Project, a fintech and training company based in Italy. Since 2019, he has also served as one of the founders and CIOs of Rocket Capital Investment, a fintech company based in Singapore.

He is the president of the Scientific Committee at SIAT and has been a member of the IFTA Board of Directors since October 2019.

Giovanni has taught the data science module in the SIAT Master's program, as well as the data science course for asset management for institutional operators.

Giovanni's articles have been featured in Milano Finanza, Traders, and Technical Analysis of Stocks & Commodities. In 2020, he published his latest book on Python and Quantitative Analysis with Hoepli.

Table of Contents

Preface

1

Introduction to Algorithmic Trading and the TradeStation Platform

Introducing algorithmic trading

Algorithmic trading definition

Algorithmic trading in quantitative hedge funds

Introducing the TradeStation platform

The TradeStation story

Download and installation procedure

Introducing TradeStation apps

Workspaces and desktops

Summary

2

Getting Hands-On with EasyLanguage

Understanding the basics of EasyLanguage

What is EasyLanguage?

How EasyLanguage works

EasyLanguage words

EasyLanguage expressions

EasyLanguage statements

EasyLanguage punctuation

Writing indicators

Exercise 01—Close

Exercise 02—Close Open

Exercise 03—Real Body

Exercise 04—MidPrice

Exercise 05—NetCh

Exercise 06—NetCh with variables

Exercise 07—Momentum

Exercise 08—Moving Average

Exercise 09—Crossover

Exercise 10—Uptrend

Exercise 11—Time

Writing strategies

How to program a basic strategy

EasyLanguage order syntax

MaxBarsBack

How to program take profit and stop loss levels

Summary

3

Writing a Trend Strategy

Developing trading ideas

Market rationale behind a trend strategy

Information

Market participants

Prices and volumes

Identifying a trend algorithmically

Moving averages

Higher highs

Handling market noise

Entry confirmations

Volatility bands

Summary

4

Strategy Backtesting and Validation

Understanding backtesting and overfitting

Strategy complexity

Strategy robustness

Sensitivity analysis

One-way sensitivity analysis

Double-way sensitivity analysis

Multiple-way sensitivity analysis

Backtesting across symbols

TradeStation stock symbol universe

ROA report in Excel

Equity lines chart on Excel

Backtesting versus buy-and-hold

In-sample, out-of-sample analysis

Modifying EasyLanguage scripts for in-sample, out-of-sample purposes

An example of in-sample, out-of-sample validation

Summary

5

Reversal Strategies

The market rationale behind a reversal strategy

Reversal up

Reversal down

Writing a reversal strategy

Existing trend

Volatility compression

Final trend

Backtesting long strategies

Identifying stock to start with

Running a multiple sensitivity analysis on HD (Home Depot Inc.)

Exporting data into Excel and creating an ROA heatmap

Selecting the best parameter set

Backtesting the strategy on the full Dow Jones 30 index

Backtesting short strategies

Running multiple-sensitivity analysis on the SPY

Exporting data into Excel and creating an ROA heatmap

Selecting the best parameter set

Summary

6

Trend Pullback Strategies

The market rationale behind a trend pullback strategy

Writing trend pullback components

Existing trend

Pullbacks

Final impulse

Assembling components

Sensitivity analysis

Out-of-sample analysis

Summary

7

Risk Management

Money management

Price-based exits

Percent-based exits

Volatility-based exits

Position sizing

The P&L equation

Equal dollar risk technique without technical exit levels

Incorporating technical exits in equal dollar risk formulas

Summary

8

Futures and Forex Algorithmic Trading

Algorithmic trading for futures markets

The basic concepts of futures

How to write algorithms for time breakout strategies

Forex algorithmic trading

The basic concepts of forex

How to write algorithms for the forex market

Breakout versus fake breakout

Summary

9

The Trading Operational Plan

What is an operational plan?

A fully automated trading plan for futures

How to manage futures symbols for real trading

How to size the four futures portfolio

How to manage live fully automated strategies

A semi-automated trading plan on large stock lists

How to track entries with RadarScreen

How to calculate volatility-based sizes

How to monitor real positions

Summary

10

EasyLanguage in AI – Bridging Traditional Trading and Advanced Analytics

Python versus EasyLanguage

Volatility Predictor on gold futures by the Gandalf Project

Bridging Python and EasyLanguage

Embedding AI predictions into EasyLanguage indicators

Embedding AI predictions into EasyLanguage strategies

Using the Volatility Predictor model as a filter

Using the Volatility Predictor model for money management

Using the Volatility Predictor model for position sizing

Creating a volatility dashboard for multiple assets

Using TradeStation to collect predictions from multiple AI models

Summary

11

EasyLanguage for Machine Learning

A definition of machine learning for pattern recognition

The Iris dataset and Fisher’s project

Labeling trading sessions

Project N.1—recognizing an up session

Selecting the target

Selecting the features

Building the pipeline

The confusion matrix

Project N.2—recognizing down sessions

Selecting the target

Selecting features

Final results

Summary

Index

Preface

Embarking on a professional journey in the world of financial market trading is like setting sail on an adventurous voyage: captivating on one side, but dangerous on the other. You will set off, or perhaps you've already set off, along with many others who share the same goal: the treasure island, symbolizing your professional success determined by financial performance. Know that very few will make it, as storms, lack of food, exhaustion, mutinies, and daily disappointments will lead the luckiest to give up before they shipwreck.

Along the journey, you might encounter pirates and sharks—people who will try to exploit your good faith. Ignore them and stay on your course, following your map. This book will teach you how to build your own personal map to navigate because having an action plan during the inevitable financial and psychological drawdowns will save your life.

Try not to work alone because, on such a difficult journey, you'll need trustworthy companions by your side. Remember that software, algorithms, and AI models are just tools—don't fall in love with them, and be ready to change them if you realize they're not helping you achieve your true goal: navigating the markets with peace of mind.

> ***NOTE***
>
> *EasyLanguage® is a registered trademark and every use of this word EasyLanguage is a registered trademark of TradeStation Technologies, Inc. All product names, logos, and brands mentioned in this book are the property of their respective owners.*

Who this book is for

Tailored for individual traders with over a year’s experience in discretionary trading, with no programming skills, this book is designed for those who’ve grappled with market losses and the inundation of trading theories lacking statistical backing. Dive into the realm of institutional insights, satisfying the hunger to learn real-world algorithmic trading applications from the institutional perspective.

This book might also interest experienced traders and fund managers who are curious to gain another perspective on the market, with a particular focus on AI integration with TradeStation and machine learning for pattern recognition.

What this book covers

Chapter 1, *Introduction to Algorithmic Trading and the TradeStation Platform*, aims to define what is meant by algorithmic trading in this book and to introduce the TradeStation platform features.

Chapter 2, *Getting Hands-On with EasyLanguage*, provides a basic EasyLanguage training course.

Chapter 3, *Writing a Trend Strategy*, shows you how to develop a trend-following strategy in the stock market, providing the rationale behind trends, a strategy development method, and the programming code.

Chapter 4, *Strategy Backtesting and Validation*, provides the methodology for backtesting and validating algorithmic strategies using EasyLanguage and TradeStation.

Chapter 5, *Reversal Strategies*, demonstrates how to develop a reversal strategy in the stock market, providing the rationale behind reversals, a strategy development method, and the programming code.

Chapter 6, *Trend Pullback Strategies*, explains how to use the techniques learned in *Chapters 3* and *5* to merge the two methodologies in order to create Trend Pullback trading strategies.

Chapter 7, *Risk Management*, addresses the critical aspect of managing financial risk in trading. While previous chapters have focused on identifying optimal entry points for trades, this chapter emphasizes the importance of managing risk through effective exits and position sizing. Given the inherent unpredictability of financial markets, controlling how much one is willing to risk is paramount.

Chapter 8, *Futures and Forex Algorithmic Trading*, explains how Futures and Forex offer numerous trading opportunities on uncorrelated instruments, allowing for both long and short positions; this chapter demonstrates how to use TradeStation and EasyLanguage for algorithmic trading in this vast market.

Chapter 9, *The Trading Operational Plan*, provides a step-by-step guide on how to set up TradeStation to create and execute an operational trading plan.

Chapter 10, *EasyLanguage in AI – Bridging Traditional Trading and Advanced Analytics*, explains how to integrate TradeStation with various AI models, creating dashboards that combine proven market monitoring and trading techniques with advanced AI tools.

Chapter 11, *EasyLanguage for Machine Learning*, explains how to program EasyLanguage for machine learning in pattern recognition, by exploring one of the earliest supervised machine learning experiments conducted by the British scientist Sir R. Fisher in the 1930s.

To get the most out of this book

To make the most of this book, it is advisable to have a basic understanding of financial markets, including concepts such as stocks, futures, Forex, and the fundamental dynamics of trading and stock charts. It is advisable to have a basic knowledge of Microsoft Excel, such as copy and paste operations, inserting simple formulas, and using pivot tables and charts, as well as basic concepts in mathematics and statistics, such as mathematical expressions, the use of parentheses, moving averages, and standard deviation.

Software/hardware covered in the book	**Operating system requirements**
TradeStation 10.0	Windows
Microsoft Excel	Windows

If you are using the digital version of this book, we advise you to type the code yourself or access the code from the book's GitHub repository (a link is available in the next section). Doing so will help you avoid any potential errors related to the copying and pasting of code.

Download the example code files

You can download the example code files for this book from GitHub at https://github.com/PacktPublishing/Tradestation-Easylanguage-for-Algorithmic-Trading. If there's an update to the code, it will be updated in the GitHub repository.

We also have other code bundles from our rich catalog of books and videos available at https://github.com/PacktPublishing/. Check them out!

Conventions used

There are a number of text conventions used throughout this book.

`Code in text`: Indicates code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter/X handles. Here is an example: "**`entryprice`** is a reserved word that provides the entry price for the current position."

A block of code is set as follows:

```
Plot1(Open,"Open");
Plot2(Close,"Close");
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:

```
Plot1( I_OpenEquity, !( "OpenEquity") ,black ) ;
```

```
Plot2( I_ClosedEquity, !( "Equity") ,darkgreen ) ;
```

Bold: Indicates a new term, an important word, or words that you see onscreen. For instance, words in menus or dialog boxes appear in **bold**. Here is an example: “From the main menu on the chart, go to **Studies** | **Edit Strategies** | **Customize**.”

> ***TIPS OR IMPORTANT NOTES***
>
> *Appear like this.*

Get in touch

Feedback from our readers is always welcome.

General feedback: If you have questions about any aspect of this book, email us at customercare@packtpub.com and mention the book title in the subject of your message.

Errata: Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you have found a mistake in this book, we would be grateful if you would report this to us. Please visit www.packtpub.com/support/errata and fill in the form.

Piracy: If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at copyright@packt.com with a link to the material.

If you are interested in becoming an author: If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, please visit authors.packtpub.com.

Share Your Thoughts

Once you’ve read *TradeStation EasyLanguage for Algorithmic Trading*, we’d love to hear your thoughts! Please click here to go straight to the Amazon review page for this book and share your feedback.

Your review is important to us and the tech community and will help us make sure we’re delivering excellent quality content.

Download a free PDF copy of this book

Thanks for purchasing this book!

Do you like to read on the go but are unable to carry your print books everywhere?

Is your eBook purchase not compatible with the device of your choice?

Don’t worry, now with every Packt book you get a DRM-free PDF version of that book at no cost.

Read anywhere, any place, on any device. Search, copy, and paste code from your favorite technical books directly into your application.

The perks don’t stop there, you can get exclusive access to discounts, newsletters, and great free content in your inbox daily

Follow these simple steps to get the benefits:

1. Scan the QR code or visit the link below

https://packt.link/free-ebook/9781835881200

2. Submit your proof of purchase
3. That’s it! We’ll send your free PDF and other benefits to your email directly

1

Introduction to Algorithmic Trading and the TradeStation Platform

There is a lot of confusion when it comes to the term **algorithmic trading**, the tools involved, and the professionals who utilize such tools in their day-to-day activities.

In the first part of this chapter, we will provide an overview of what is meant by algorithmic trading, which tools make sense for the individual trader, and which professionals utilize algorithms in their trading process.

In the second part of the chapter, we will introduce the **TradeStation platform**, showing the installation process and its main functionalities.

In this chapter, we're going to cover the following main topics:

- Introducing algorithmic trading
- Introducing the TradeStation platform

By the end, we will have demystified the concept of algorithmic trading and you will have gained a clear understanding of the tools relevant to individual traders.

You can find all the information related to TradeStation expenses at the following link: https://www.tradestation.com/pricing/service-fees/.

Introducing algorithmic trading

There are various definitions of algorithmic trading, so we need to first clarify what we mean by this term in this book and what the perspective of these tools is from the viewpoint of professional users.

To achieve this, we are now going to cover the following:

- A definition of algorithmic trading
- What a quantitative hedge fund is from an organizational point of view

Algorithmic trading definition

According to a very common definition from Wikipedia, "Algorithmic trading involves using computer algorithms to help traders execute trading strategies based on factors like price, volume, and timing. The main advantages of algorithmic trading are speed and efficiency, backtesting capabilities, reduced emotional bias and diversification."

Such a definition reflects what most people think about algorithmic trading but it doesn't fit exactly the professional environment view. Let's see why:

- First, let's clarify **execution**. Using algorithms and automating trade execution are two different matters. We may use algorithms just to automate a research process to analyze the market or to develop tools to monitor the market. We can also use algorithms to fully auto-execute trades but, as you will learn, this is just a component of a wider trading project.
- Let's clarify the concept of **speed**. The primary goal of algorithms is not always to make things faster; rather, it's about providing support in the decision-making process and solving problems efficiently and effectively. In addition, fast trading is not suitable for individual traders, as it requires huge investments in technology and, therefore, it is a battlefield where big companies operate.

  In the coming chapters, we will learn how to choose a wiser approach for trade execution than increasing trading speed.
- The third misunderstanding is about **emotions**. At a certain point in the trading decision process, emotions will kick in. If you are a discretionary trader, emotions will kick in when you click the **Buy**/**Sell** button. If you are an algorithmic trader, they will kick in when you activate the **ON**/**OFF** system button. Emotions are just delayed, not removed. Algorithms will help you approach the market rationally and scientifically, increasing your confidence level, and thus reducing your stress during the trade, but you will never get rid of emotions.

In *Figure 1.1*, you can see how algorithmic trading tools can be classified:

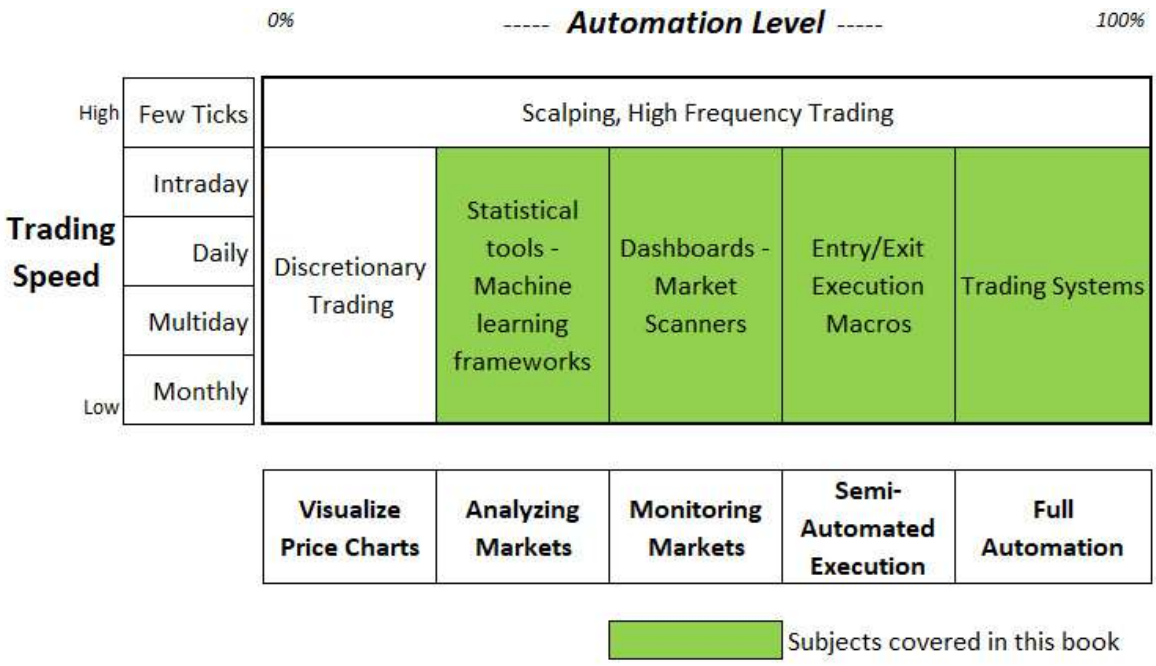


Figure 1.1 - Algorithmic trading tools by trading speed and automation level

In *Figure 1.1*, we classify algorithmic trading tools by trading speed (ranging from a few ticks to monthly time frames) and automation level (ranging from 0 to 100%).

As you can see, there are several ways to deal with such tools. Apart from discretionary trading and high-speed tools, which are not suited for individual traders, this book will help you do the following:

- Program **analysis** tools such as statistical tools and machine learning frameworks

- Program **monitoring** tools, such as real-time trading dashboards and equity market scanners
- Program **mechanical trading** tools such as execution macros
- Program **trading systems** with a full automation approach

In conclusion, when we use the term *algorithmic trading* in this book, we mean any computer-based tool able to help traders analyze, validate, monitor, and execute trading strategies.

Algorithmic trading in quantitative hedge funds

When we talk about algorithms, we often forget that they are going to be used by humans. Therefore, I think it is essential to briefly describe my experience in the finance sector, where I worked for small quantitative hedge fund start-ups.

A **quantitative hedge fund** generally originates from the idea of a trader who, after several years of experience in the markets, decides to start a hedge fund business. Subsequently, they partner with other individuals who will handle the legal, administrative, and commercial management of the company.

In this initial phase of the hedge fund start-up, it is generally not a priority to build a proprietary trading platform. Hence, such funds find it very useful to utilize platforms such as TradeStation, which are ready for use. So, these funds generally temporarily hire a specialist familiar with EasyLanguage (TradeStation's programming language) to transform the senior trader's market insights into algorithmic tools that can help the entire team in the trading process. Experienced traders generally have a visual approach to the market, relying on charts as their primary tools, which they have been using for decades. Therefore, the EasyLanguage specialist is partnered with the traders for a certain period, seeks to align with their approach, and then constructs trading tools based on that understanding. Depending on the trading strategy, sometimes such funds hire a traders' team: let's imagine a trading room with 5–10 workstations, where traders work following the same dashboard that provides the team with the market perspective of the head trader. Simultaneously, they may have the freedom to act within well-defined risk rules. In other words, this type of organization leverages the computational power of algorithms combined with the individual sensitivity of the human trader.

Some other funds are a *one-man band*, and the trading decision process is very similar to the individual trader's—except for the investors' pressure and compliance matters.

At times, these funds create a small development department to continue research while the rest of the organization operates in the market.

In this way, TradeStation's EasyLanguage becomes an indispensable tool both in research and development activities and in daily trading operations.

Introducing the TradeStation platform

TradeStation is a financial brokerage and trading platform that allows users to trade various financial instruments. Key features of TradeStation typically include advanced charting and technical analysis tools, customizable trading strategies, and a variety of order types. It is known for its focus on providing tools for technical analysis and algorithmic trading.

Let's get an overview of TradeStation by going through the following:

- The TradeStation story
- The download and installation procedure
- TradeStation apps
- **The working environment**: Workspaces and desktops

The TradeStation story

When I started using TradeStation and EasyLanguage many years ago, I had the impression that the platform and the language editor understood what I, as a trader, was looking for. It was a bit like using a co-pilot, but at that time, AI was just a movie.

Now, there are many software options available for algorithmic trading, but at an early stage as a programmer, TradeStation's EasyLanguage (or PowerLanguage from MultiCharts, which is 90% supported) is the best choice because it doesn't require previous programming experience and has a short learning curve.

MULTICHARTS POWERLANGUAGE

MultiCharts *(www.multicharts.com) is a trading platform very similar to TradeStation. The main differences are in the broker integration both for trading and data feeding, as MultiCharts supports integration with various brokers, and TradeStation is both a trading platform and a brokerage. All the EasyLanguage scripts provided in this book are 100% supported by MultiCharts.*

TradeStation has quite a compelling story in the world of trading and financial technology. Originally founded in 1982 by William and Rafael Cruz, TradeStation started as Omega Research, Inc., focusing on developing tools for traders. In 1991, they launched the TradeStation platform, which quickly gained attention for its advanced charting, technical analysis tools, and strategy backtesting capabilities.

As the years progressed, TradeStation evolved into a comprehensive trading platform offering not just analysis tools but also order execution capabilities for equities, options, futures, and forex

trading. It became known for its robustness in catering to active and professional traders, offering customizable features and algorithmic trading capabilities.

In 2011, TradeStation was acquired by Monex Group, a global financial services company. This acquisition further expanded TradeStation's reach and resources, allowing for enhancements in technology and service offerings.

TradeStation has continued to innovate, adapting to changing market needs and technological advancements. It has remained a prominent player in the trading platform arena, often favored by traders seeking powerful analytical tools and a comprehensive trading environment.

The platform's journey has been characterized by a commitment to providing traders with cutting-edge technology, a user-friendly interface, and a suite of tools to support their trading strategies across various financial markets.

Now that you know the story of TradeStation, let's download and install it on your system.

Download and installation procedure

First of all, you need to open a TradeStation account at http://www.tradestation.com/ and follow the instructions provided on the main web page.

Once you have an account, you can log in to the website and download the desktop platform by going to **Download Software** | **Download Platform**. Then, follow the straightforward installation instructions.

There are some prerequisites for it, which are listed next. Please check the minimum requirements before installation:

Processor	**Dual-core Intel® or AMD® processor at 1.5 GHz or faster**
Memory (RAM)	8 GB
Hard Drive	5,400 RPM drive 400 MB free space or more
Video Cards	32-bit graphics support 256 MB for a single monitor
Screen Resolution	1280×1024 pixels or higher

Processor	**Dual-core Intel® or AMD® processor at 1.5 GHz or faster**
Operating Systems	Windows® 10 64-bit Windows® 10 32-bit
Internet Browser	Microsoft Edge browser
Internet Connection	Broadband connection 2 Mbps or better (download)

Table 1.1 - Technical minimum requirements

In case you need any kind of technical help related to the platform, you can find a very active community at https://community.tradestation.com/discussions/.

Once the installation is complete, double-click on the TradeStation icon that you will find on your desktop. The platform will look like *Figure 1.2*. Please note that every time we mention the main menu, we are referring to the menu you see in *Figure 1.2*:

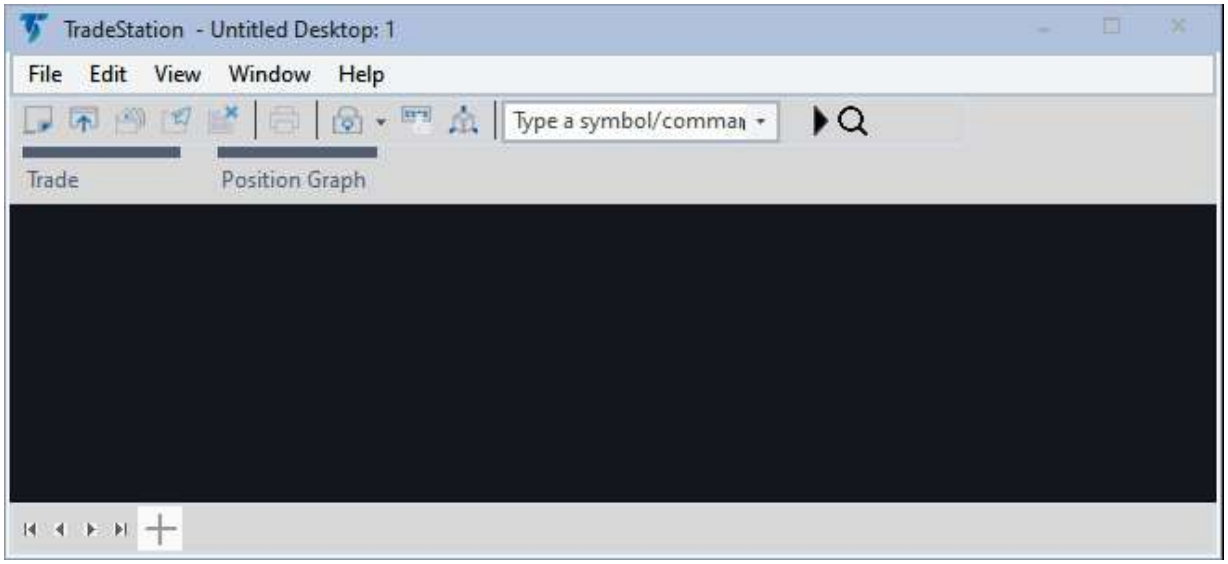


Figure 1.2 - Main menu

We are going now to introduce the main TradeStation apps.

Introducing TradeStation apps

The latest release of the TradeStation platform (version 10, update 72) is made of 20 apps. by clicking on **File** | **New Application**, we can see the list of available applications (*Figure 1.3*):

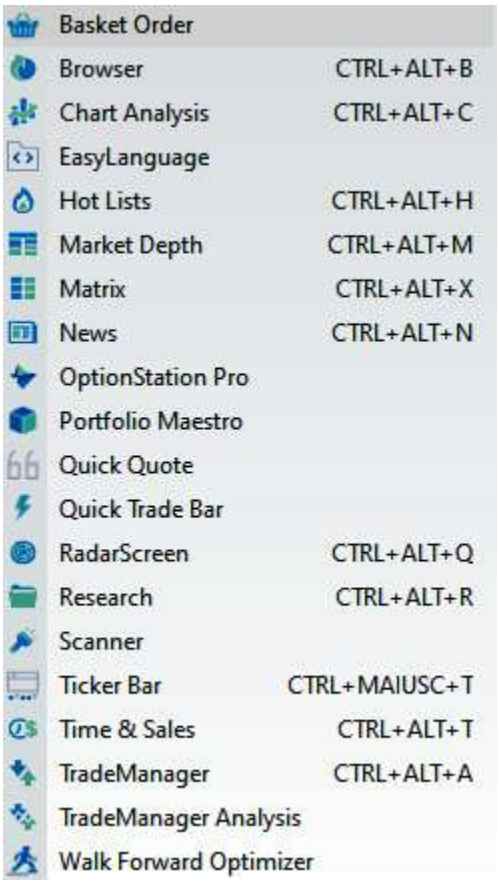


Figure 1.3 - TradeStation apps

In *Figure 1.3*, you see the 20 applications available listed in alphabetical order. For the algorithmic training purpose of this book, we are mainly going to deal with four of them:

- **Chart Analysis**: We can use this to visualize price charts and indicators or strategies applied to them
- **RadarScreen**: We can use this to monitor up to 1,000 markets in real time
- **Scanner**: We can use this to scan the entire symbols universe for specific trading opportunities based on predefined criteria
- **EasyLanguage**: We can use this to manage the **TradeStation Development Environment** page (or EasyLanguage editor)

Chart analysis

A **chart analysis** is a chart that visually represents the historical price and volume data of a security over a specific period. It is the basic brick where you can add indicators or strategies; such functionalities will be illustrated starting in *Chapter 2*.

To create a chart analysis from the main menu, go to **File** | **New Application** | **Chart Analysis**. In *Figure 1.4*, you see a chart for **SPY**, which is the financial asset representing the main 500 US companies:

Figure 1.4 - Chart analysis for SPY ETF

As you can see in *Figure 1.4*, the chart settings can be modified by the menu available on top of the chart, where you can manage the following:

- Timeframe
- Drawing
- Studies
- Strategies
- Data
- Style
- Settings

To choose the chart's timeframe, click on **Timeframe**, and then select your preferred one, as shown in *Figure 1.5*. For example, in *Figure 1.4*, we used the **Daily** timeframe option:

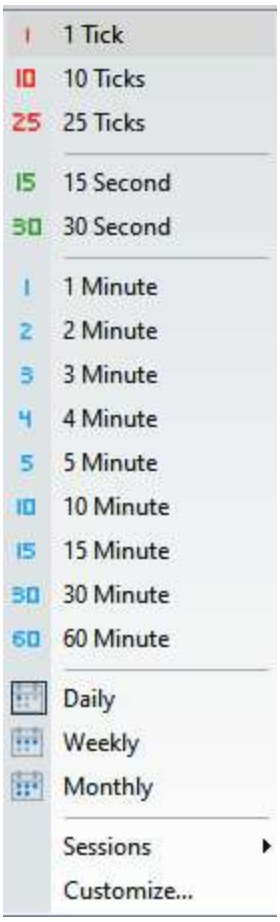


Figure 1.5 - Chart timeframe

The **Drawing** menu allows you to incorporate drawings into the chart. However, this feature is seldom utilized in algorithmic trading.

To apply a study, from the chart menu, go to **Studies** | **Add Study**. A list similar to *Figure 1.6* will show you the studies available: all your custom studies will be shown in this list.

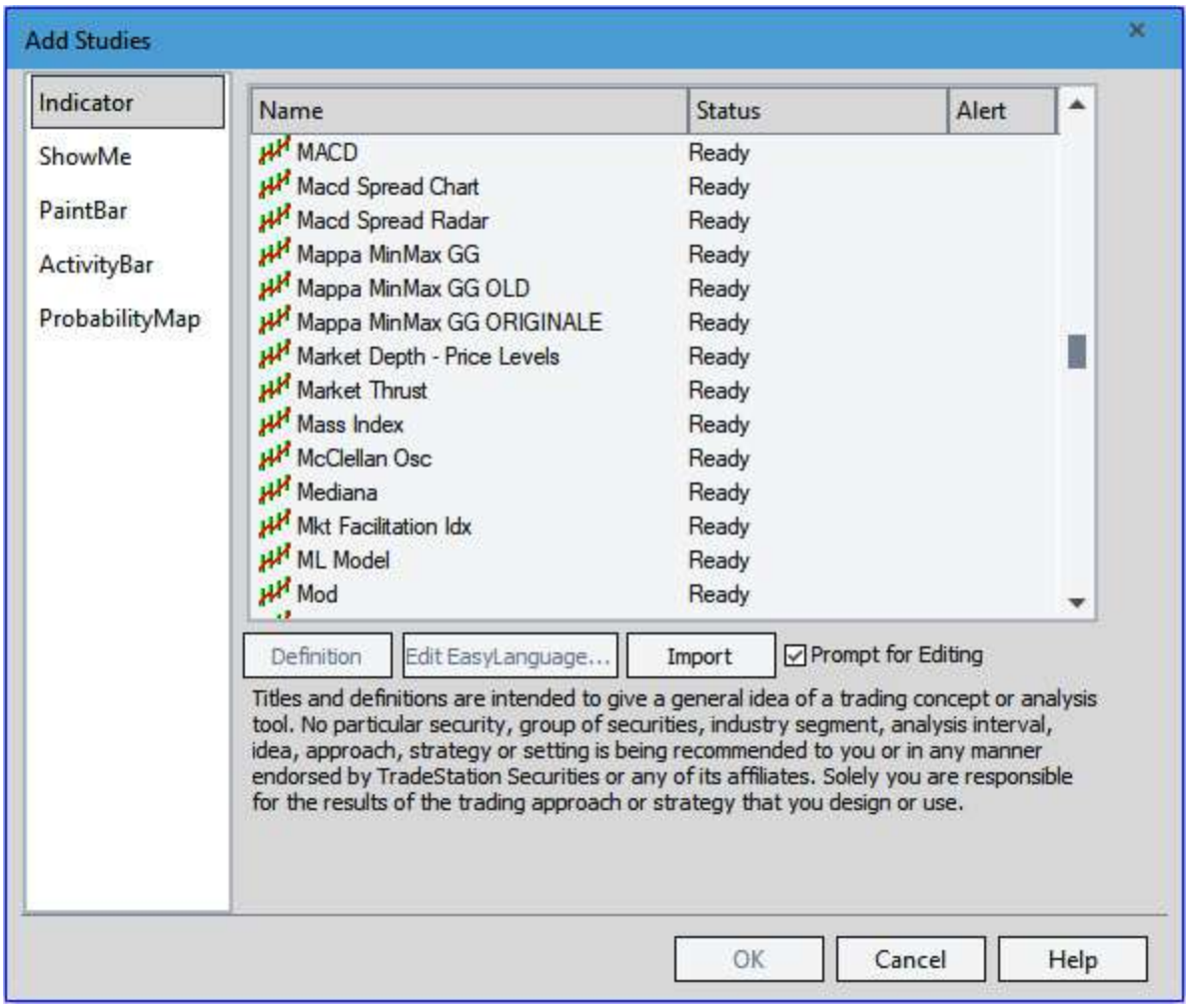


Figure 1.6 - Chart studies

For example, in *Figure 1.7*, we applied a **MACD** indicator on the **Daily** chart of SPY:

Figure 1.7 - MACD indicator

To apply a strategy, from the chart menu, go to **Studies** | **Add Strategy**. A list similar to *Figure 1.8* will show you the strategies available: all your custom strategies will be shown in this list.

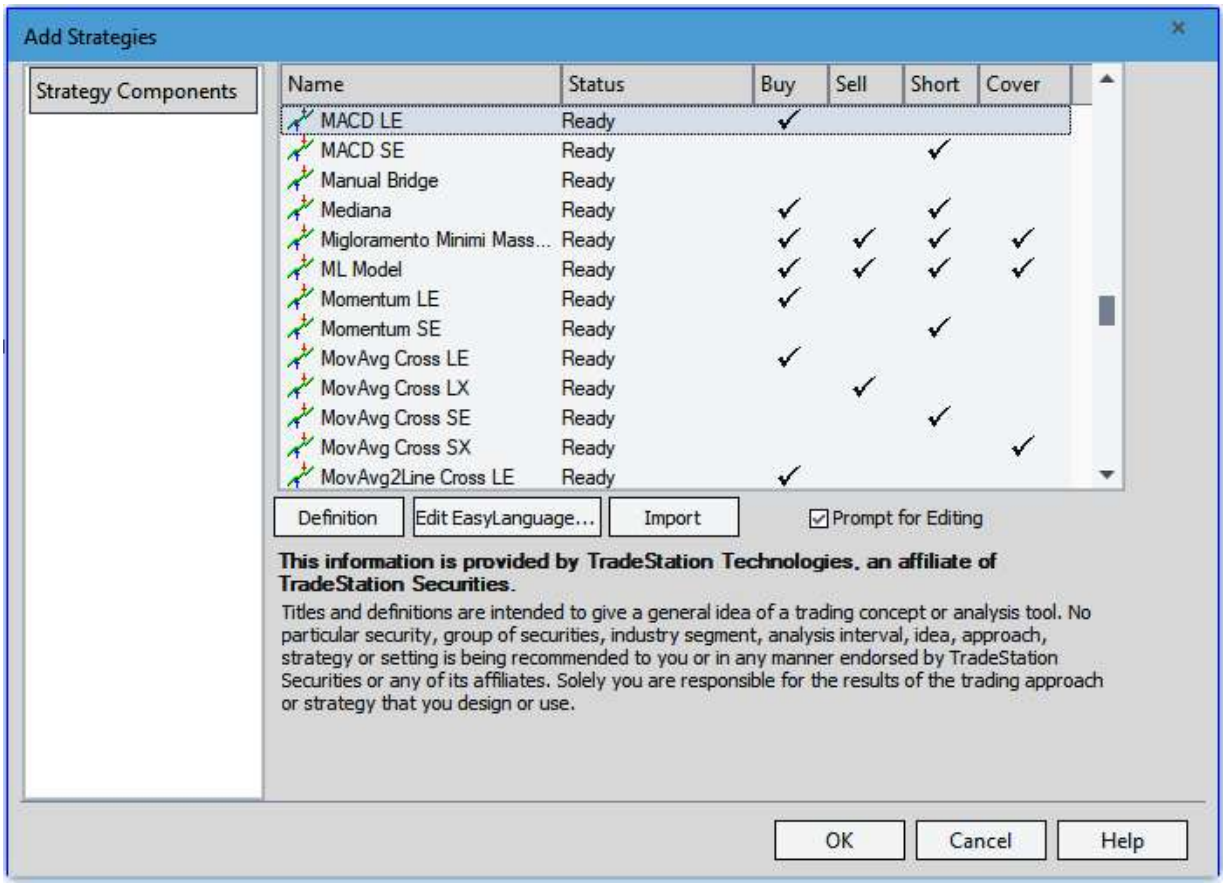

Add Strategies
Strategy Components
Name
Status
Buy
Sell
Short
Cover
MACD LE Ready
MACD SE Ready
Manual Bridge Ready
Mediana Ready
Migloramento Minimi Mass... Ready
ML Model Ready
Momentum LE Ready
Momentum SE Ready
MovAvg Cross LE Ready
MovAvg Cross LX Ready
MovAvg Cross SE Ready
MovAvg Cross SX Ready
MovAvg2Line Cross LE Ready
Definition
Edit EasyLanguage...
Import
Prompt for Editing
This information is provided by TradeStation Technologies, an affiliate of TradeStation Securities.
Titles and definitions are intended to give a general idea of a trading concept or analysis tool. No particular security, group of securities, industry segment, analysis interval, idea, approach, strategy or setting is being recommended to you or in any manner endorsed by TradeStation Securities or any of its affiliates. Solely you are responsible for the results of the trading approach or strategy that you design or use.
OK
Cancel
Help

Figure 1.8 - Chart strategies

For example, in *Figure 1.9*, we added a **MACD** strategy on the same **Daily** chart of SPY:

Figure 1.9 - MACD strategy

To manage the historical data or change the symbol properties, you can click on **Data** and navigate the list shown in *Figure 1.10*:

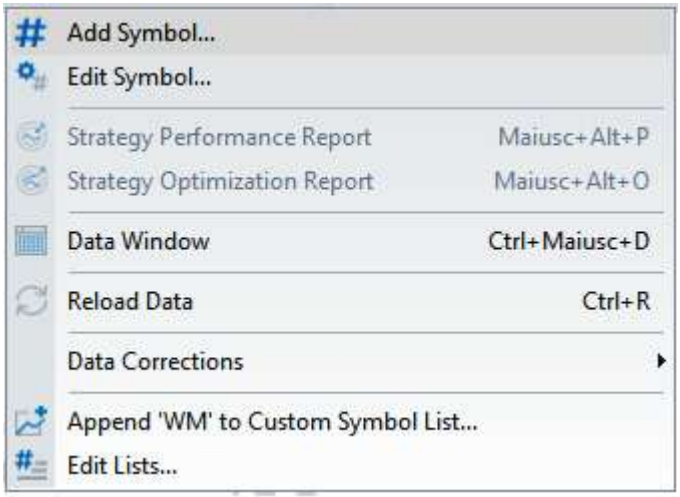


Figure 1.10 - Chart data

To manage the chart's style, you can click on **Style** and navigate the list shown in *Figure 1.11*:

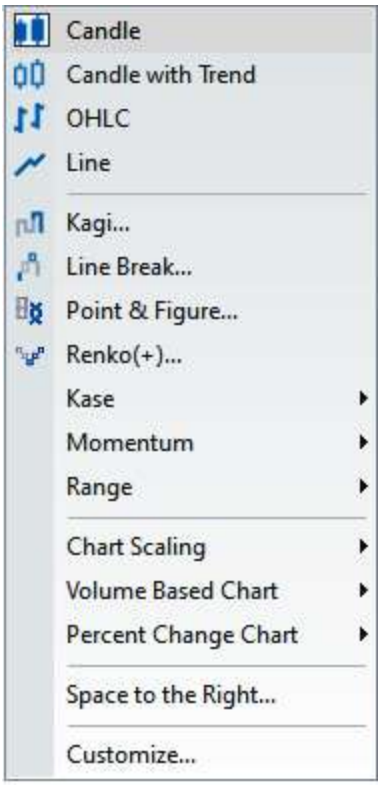


Figure 1.11 - Chart styles

If you click on **Settings**, you can find additional resources to customize your chart (*Figure 1.12*):

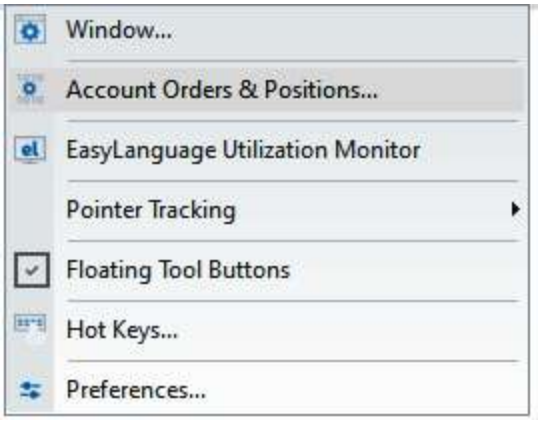


Figure 1.12 - Chart settings

We now understand how to manage TradeStation charts. Being familiar with them is crucial for algorithm programming because the charts are generated from the same datasets that will be used for creating both indicators and strategies. As you will see in the coming chapters, charts will be the primary tool you use to determine whether the strategies you have written are correct.

RadarScreen

RadarScreen is a real-time scanning and ranking tool that allows you to apply technical and fundamental indicators to a list of symbols in a tabular format. Each symbol row in RadarScreen is similar to a chart in that you can access historical data at any bar interval for any symbol as needed for your indicator calculations. There are hundreds of indicator columns covering all aspects of market analysis that can be inserted into RadarScreen. You can build symbol watch lists by inserting pre-built symbol lists or you can create your own custom lists. RadarScreen can dynamically sort and rank groups of symbols based on alert criteria, price information, or indicator calculations, and you can customize any calculation or alert for the way you trade.

To create a **RadarScreen** page, go to **File** |**New Application** | **RadarScreen** or press *Ctrl + Alt + Q*. The **RadarScreen** page will appear as in *Figure 1.13*:

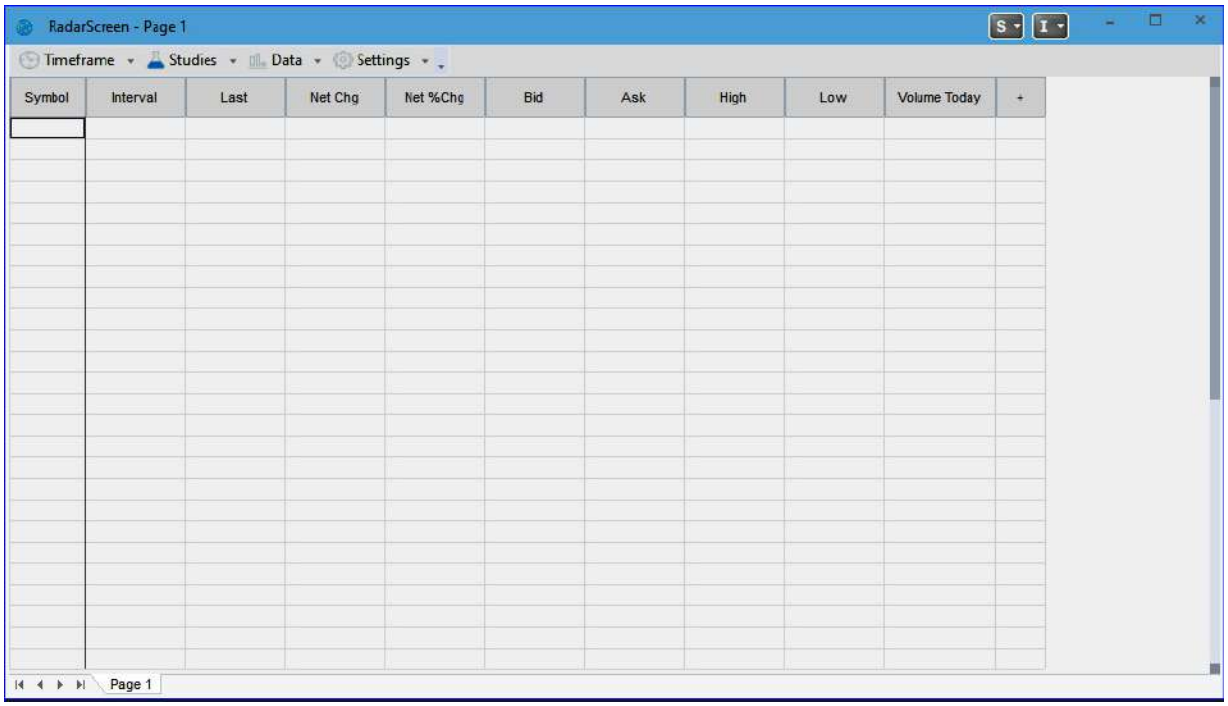


Figure 1.13 - RadarScreen

There are two ways to insert symbols into a RadarScreen page.

The first way is to insert symbols manually by just typing the ticker into any cell in the **`Symbol`** column (*Figure 1.14*):

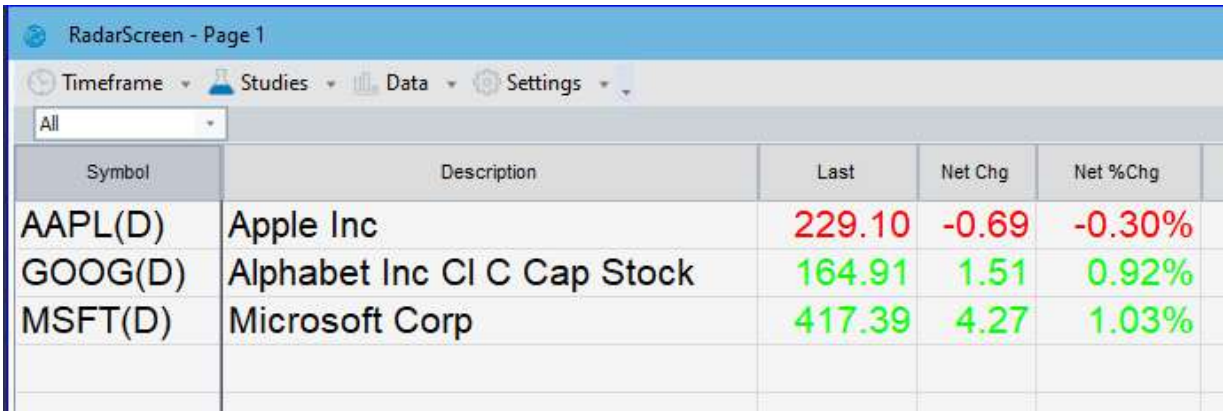


Figure 1.14 - RadarScreen—adding a symbol manually

Please note that on **RadarScreen** pages, you can manually edit just the symbol names.

The second way is to go to **Data** | **Add Symbol List** and choose your preferred symbol list among the many TradeStation lists available (*Figure 1.15*):

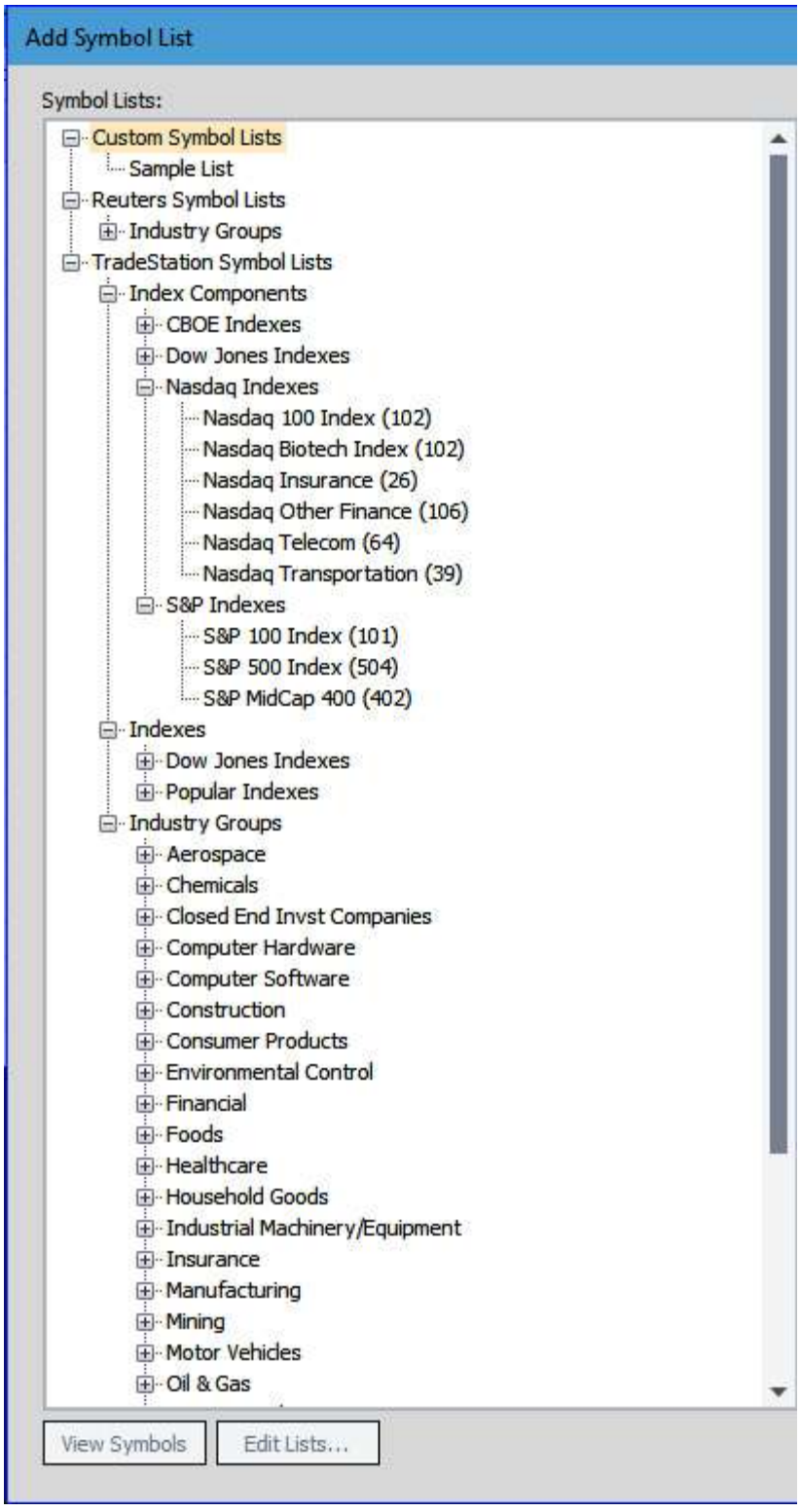
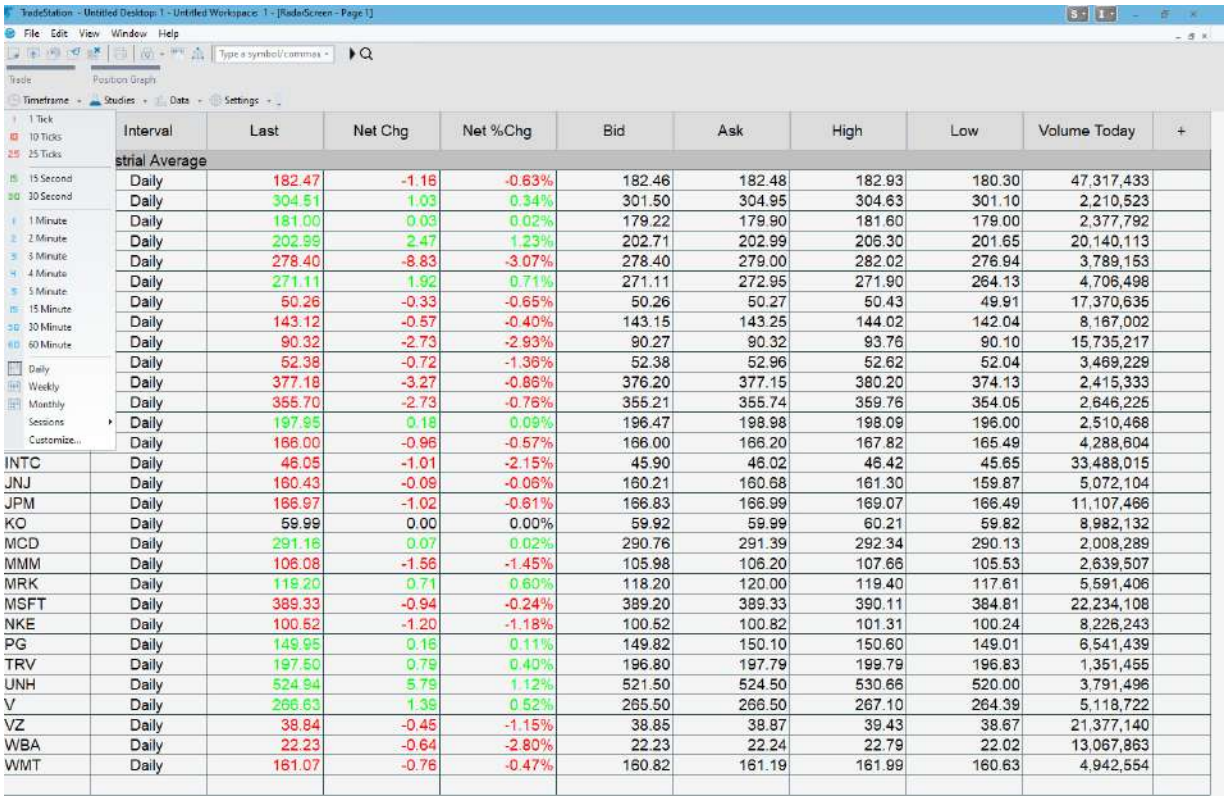


Figure 1.15 - RadarScreen—TradeStation symbol lists

For example, in *Figure 1.16*, we chose the **Dow Jones 30** list:



	Interval	Last	Net Chg	Net %Chg	Bid	Ask	High	Low	Volume Today	+
strial Average										
	Daily	182.47	-1.16	-0.63%	182.46	182.48	182.93	180.30	47,317,433	
	Daily	304.51	1.03	0.34%	301.50	304.95	304.63	301.10	2,210,523	
	Daily	181.00	0.03	0.02%	179.22	179.90	181.60	179.00	2,377,792	
	Daily	202.99	2.47	1.23%	202.71	202.99	206.30	201.65	20,140,113	
	Daily	278.40	-8.83	-3.07%	278.40	279.00	282.02	276.94	3,789,153	
	Daily	271.11	1.92	0.71%	271.11	272.95	271.90	264.13	4,706,498	
	Daily	50.26	-0.33	-0.65%	50.26	50.27	50.43	49.91	17,370,635	
	Daily	143.12	-0.57	-0.40%	143.15	143.25	144.02	142.04	8,167,002	
	Daily	90.32	-2.73	-2.93%	90.27	90.32	93.76	90.10	15,735,217	
	Daily	52.38	-0.72	-1.36%	52.38	52.96	52.62	52.04	3,469,229	
	Daily	377.18	-3.27	-0.86%	376.20	377.15	380.20	374.13	2,415,333	
	Daily	355.70	-2.73	-0.76%	355.21	355.74	359.76	354.05	2,646,225	
	Daily	197.95	0.18	0.09%	196.47	198.98	198.09	196.00	2,510,468	
	Daily	166.00	-0.96	-0.57%	166.00	166.20	167.82	165.49	4,288,604	
INTC	Daily	46.05	-1.01	-2.15%	45.90	46.02	46.42	45.65	33,488,015	
JNJ	Daily	160.43	-0.09	-0.06%	160.21	160.68	161.30	159.87	5,072,104	
JPM	Daily	166.97	-1.02	-0.61%	166.83	166.99	169.07	166.49	11,107,466	
KO	Daily	59.99	0.00	0.00%	59.92	59.99	60.21	59.82	8,982,132	
MCD	Daily	291.16	0.07	0.02%	290.76	291.39	292.34	290.13	2,008,289	
MMM	Daily	106.08	-1.56	-1.45%	105.98	106.20	107.66	105.53	2,639,507	
MRK	Daily	119.20	0.71	0.60%	118.20	120.00	119.40	117.61	5,591,406	
MSFT	Daily	389.33	-0.94	-0.24%	389.20	389.33	390.11	384.81	22,234,108	
NKE	Daily	100.52	-1.20	-1.18%	100.52	100.82	101.31	100.24	8,226,243	
PG	Daily	149.95	0.16	0.11%	149.82	150.10	150.60	149.01	6,541,439	
TRV	Daily	197.50	0.79	0.40%	196.80	197.79	199.79	196.83	1,351,455	
UNH	Daily	524.94	5.79	1.12%	521.50	524.50	530.66	520.00	3,791,496	
V	Daily	266.63	1.38	0.52%	265.50	266.50	267.10	264.39	5,118,722	
VZ	Daily	38.84	-0.45	-1.15%	38.85	38.87	39.43	38.67	21,377,140	
WBA	Daily	22.23	-0.64	-2.80%	22.23	22.24	22.79	22.02	13,067,863	
WMT	Daily	161.07	-0.76	-0.47%	160.82	161.19	161.99	160.63	4,942,554	

Figure 1.16 - RadarScreen Dow Jones 30

To choose the RadarScreen timeframe, you need to click on **Timeframe** and select your preferred one. The list is the same as we saw for Chart Analysis.

To add indicators, from the **RadarScreen** menu, go to **Studies** and then select your preferred one from the list of available indicators.

From the **Settings** menu, you can customize the layout of the RadarScreen tool, such as managing the page settings, captions, and templates (*Figure 1.17*):

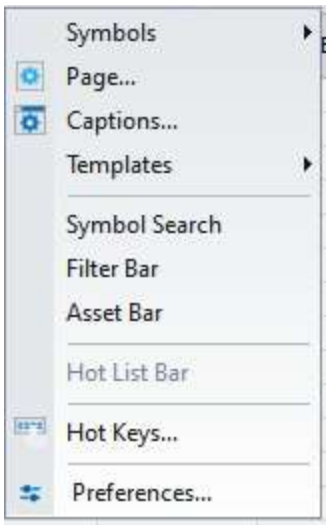


Figure 1.17 - RadarScreen settings

From **Settings** | **Page**, you can manage settings such as **Color**, **Font**, and **Grid** (*Figure 1.18*):

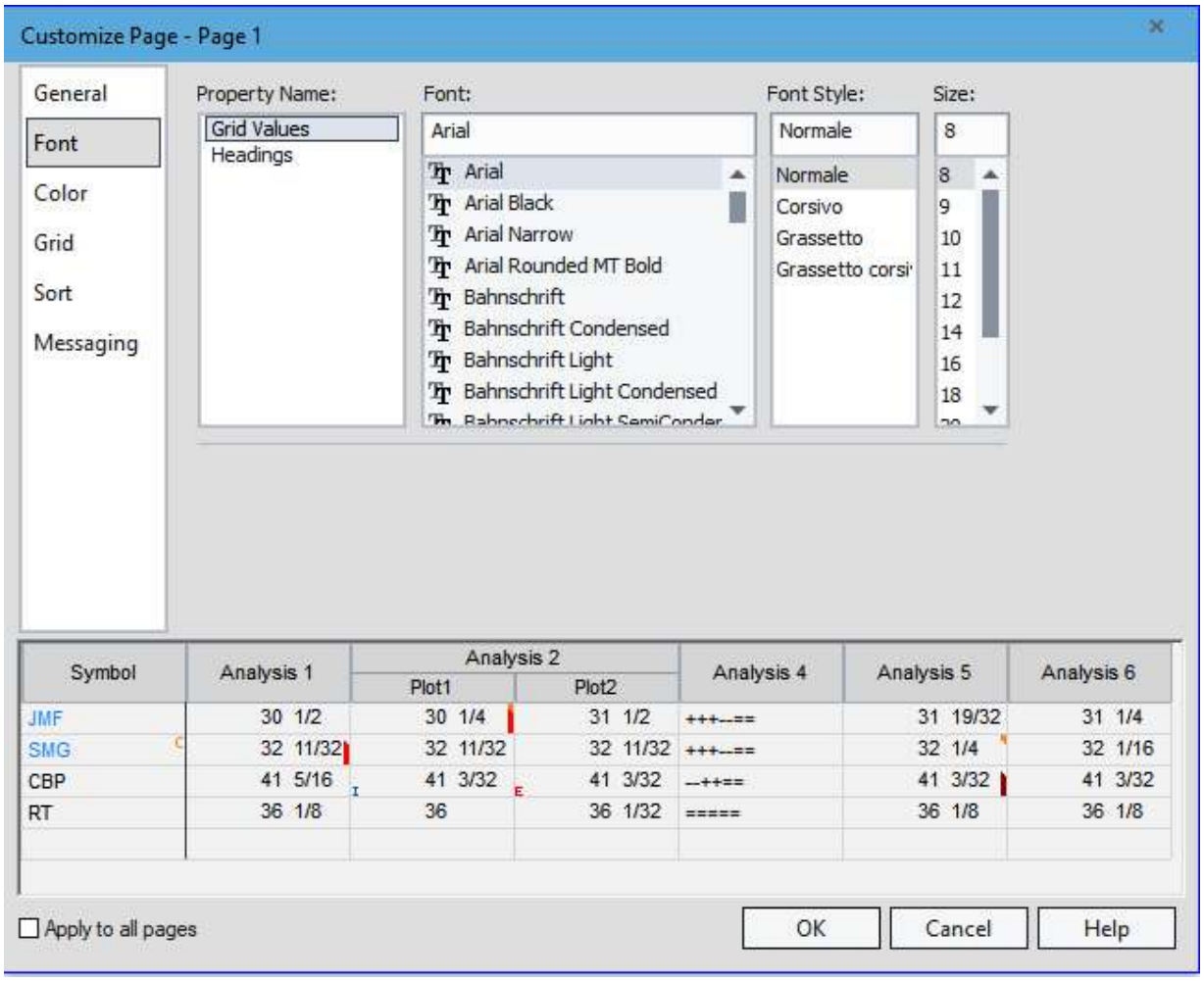


Figure 1.18 - Settings | Page

Using RadarScreen is similar to having the big picture of multiple charts showing just the last real-time reading both for the price and for any indicator applied. Please remember that RadarScreen has a limitation of 1,000 rows; if you need to manage longer lists, you may need to use Scanner.

Scanner

The **Scanner** app is a tool to scan lists with more than 1,000 symbols. It is not in real time, and it is very useful for equity traders interested in daily/weekly/monthly market analysis.

You can open an empty **Scanner** page from the main menu by going to **File** | **New Application** | **Scanner**. It will appear as shown in *Figure 1.19*:

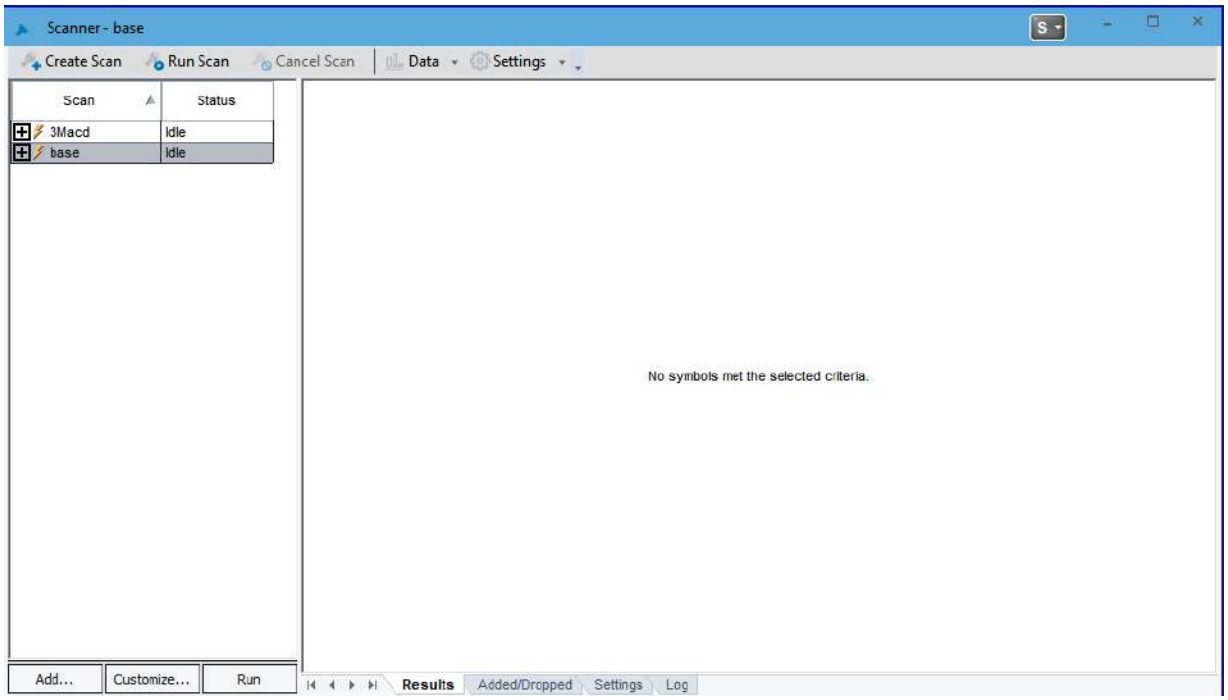


Figure 1.19 - Empty Scanner page

Scanner pages are a must when you deal with thousands of stock symbols. We will see how to set and use Scanner in *Chapter 3*.

EasyLanguage

From the main menu, go to **File** | **New Application** | **EasyLanguage** to open the **TradeStation Development Environment** page (or EasyLanguage editor). It will appear as an empty space with a menu, as shown in *Figure 1.20*. Every time we talk about the EasyLanguage menu, we will be referring to this:

Figure 1.20 - TradeStation Development Environment (or EasyLanguage editor)

Try to create a new indicator from the EasyLanguage editor menu. Go to **File** | **New** | **Indicator**, give it the name `myindicator`, and then click **OK**:

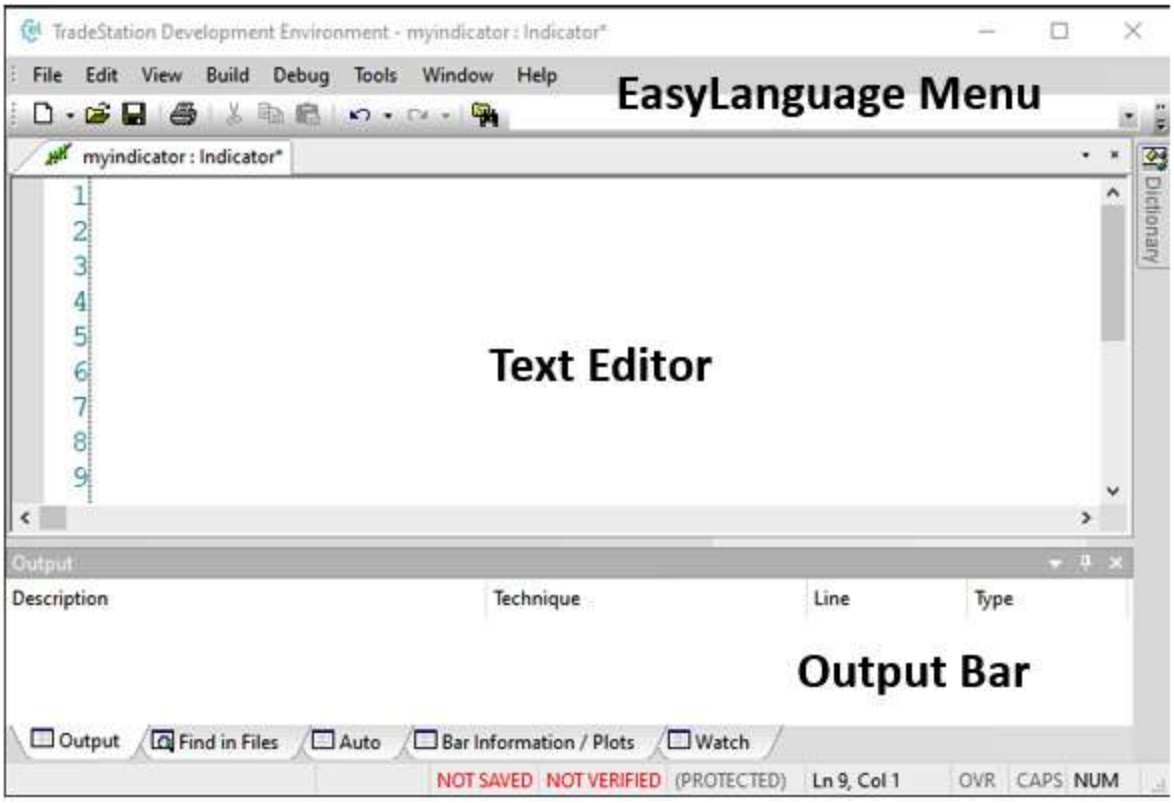


Figure 1.21 - The myindicator indicator

In *Figure 1.21*, you can see what the editor will look like. For the purpose of this book, you are mainly going to use the text editor section, in which you will write 100% of your code, and the output bar, where compiling errors will appear.

Workspaces and desktops

TradeStation allows the creation of customized working environments by using the following:

- Workspaces
- Desktops

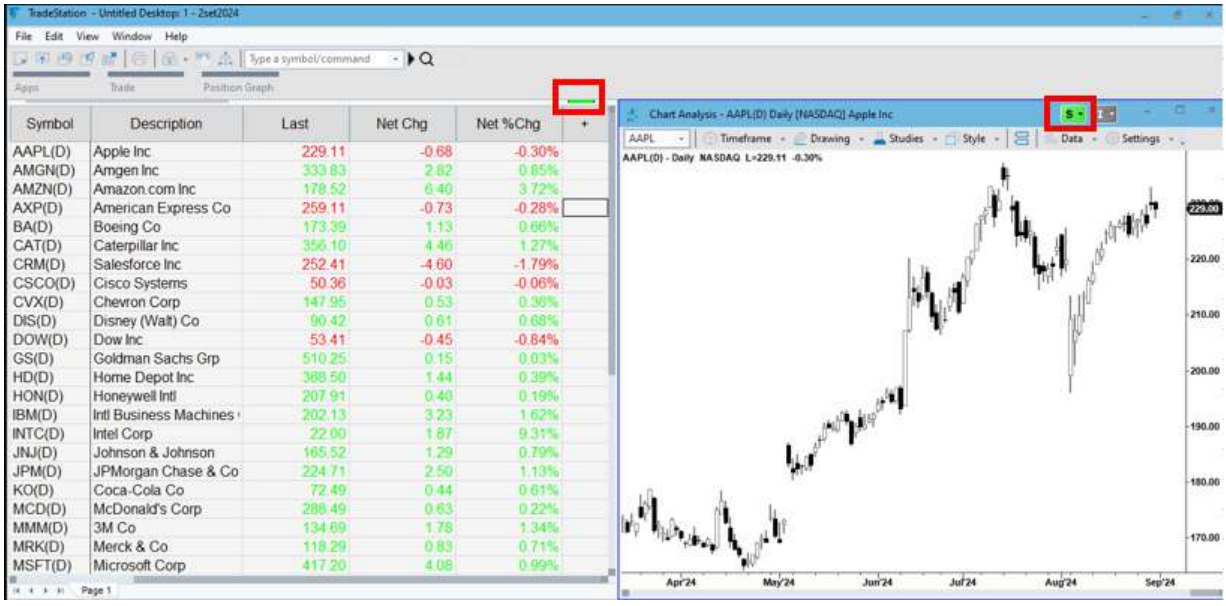
Workspaces

A **workspace** is a customizable environment where users can arrange and organize different apps such as Chart Analysis, RadarScreen, and other analysis tools. The workspace allows traders to have a tailored setup that suits their trading preferences and strategies.

In *Figure 1.22*, you can see a workspace populated with multiple charts:

Figure 1.22 - Workspace with multiple charts

In *Figure 1.23*, you can see a workspace containing one RadarScreen linked to one chart:



Symbol	Description	Last	Net Chg	Net %Chg
AAPL(D)	Apple Inc	229.11	-0.68	-0.30%
AMGN(D)	Amgen Inc	333.83	2.82	0.85%
AMZN(D)	Amazon.com Inc	178.52	6.40	3.72%
AXP(D)	American Express Co	259.11	-0.73	-0.28%
BA(D)	Boeing Co	173.39	1.13	0.66%
CAT(D)	Caterpillar Inc	356.10	4.46	1.27%
CRM(D)	Salesforce Inc	252.41	-4.60	-1.79%
CSCO(D)	Cisco Systems	50.36	-0.03	-0.06%
CVX(D)	Chevron Corp	147.95	0.53	0.36%
DIS(D)	Disney (Walt) Co	90.42	0.61	0.68%
DOW(D)	Dow Inc	53.41	-0.45	-0.84%
GS(D)	Goldman Sachs Grp	510.25	0.15	0.03%
HD(D)	Home Depot Inc	368.50	1.44	0.39%
HON(D)	Honeywell Intl	207.91	0.40	0.19%
IBM(D)	Intl Business Machines	202.13	3.23	1.62%
INTC(D)	Intel Corp	22.00	1.87	9.31%
JNJ(D)	Johnson & Johnson	165.52	1.29	0.79%
JPM(D)	JPMorgan Chase & Co	224.71	2.50	1.13%
KO(D)	Coca-Cola Co	72.49	0.44	0.61%
MCD(D)	McDonald's Corp	288.49	0.63	0.22%
MMM(D)	3M Co	134.69	1.78	1.34%
MRK(D)	Merck & Co	118.29	0.83	0.71%
MSFT(D)	Microsoft Corp	417.20	4.08	0.99%

Figure 1.23 - Workspace with RadarScreen and chart linked

Remember that to link a chart to a RadarScreen, you need to give the **Symbol Link** button the same color, as shown in *Figure 1.24*:

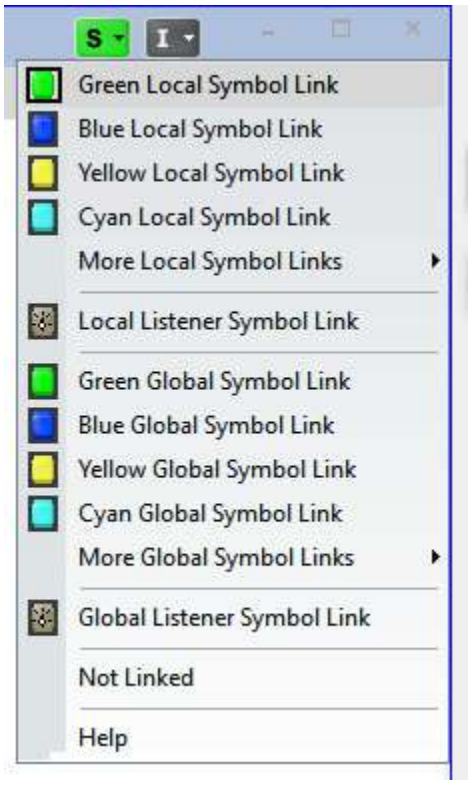


Figure 1.24 - Symbol Link

Workspaces are a very useful tool to customize your working environment according to your needs. Remember that your needs may change during the week and/or during the day; for example, you might use a workspace with Scanner while markets are closed and a workspace with RadarScreen and Chart Analysis during the trading sessions. Workspaces can be managed by going to **File** | **Workspace**.

Multiple workspaces can be collected into one desktop.

Desktops

Desktops are very useful if you manage multiple monitors: for example, if you have two monitors, you could open one desktop dedicated to equities on monitor 1 and a second desktop dedicated to forex on monitor 2.

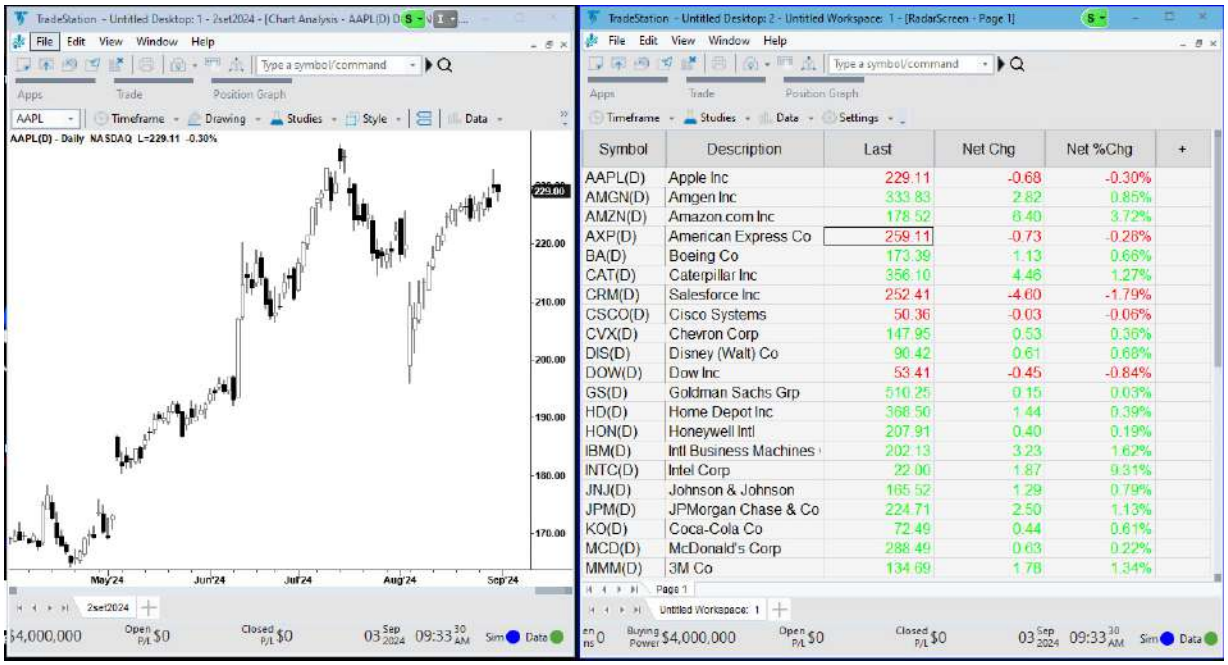

TradeStation - Untitled Desktop: 1 - 2set2024 - [Chart Analysis - AAPL(D) D
File Edit View Window Help
Type a symbol/command
Apps Trade Position Graph
AAPL Timeframe Drawing Studies Style Data
AAPL(D) - Daily NASDAQ L=229.11 -0.30%
229.00
220.00
210.00
200.00
190.00
180.00
170.00
May'24 Jun'24 Jul'24 Aug'24 Sep'24
2set2024
$4,000,000 Open P/L $0 Closed P/L $0 03 Sep 2024 09:33 30 AM Sim Data
TradeStation - Untitled Desktop: 2 - Untitled Workspace: 1 - [RadarScreen - Page 1]
File Edit View Window Help
Type a symbol/command
Apps Trade Position Graph
Timeframe Studies Data Settings
Symbol Description Last Net Chg Net %Chg +
AAPL(D) Apple Inc 229.11 -0.68 -0.30%
AMGN(D) Amgen Inc 333.83 2.82 0.85%
AMZN(D) Amazon.com Inc 178.52 6.40 3.72%
AXP(D) American Express Co 259.11 -0.73 -0.28%
BA(D) Boeing Co 173.39 1.13 0.66%
CAT(D) Caterpillar Inc 356.10 4.46 1.27%
CRM(D) Salesforce Inc 252.41 -4.60 -1.79%
CSCO(D) Cisco Systems 50.36 -0.03 -0.06%
CVX(D) Chevron Corp 147.95 0.53 0.36%
DIS(D) Disney (Walt) Co 90.42 0.61 0.68%
DOW(D) Dow Inc 53.41 -0.45 -0.84%
GS(D) Goldman Sachs Grp 510.25 0.15 0.03%
HD(D) Home Depot Inc 368.50 1.44 0.39%
HON(D) Honeywell Intl 207.91 0.40 0.19%
IBM(D) Intl Business Machines 202.13 3.23 1.62%
INTC(D) Intel Corp 22.00 1.87 9.31%
JNJ(D) Johnson & Johnson 165.52 1.29 0.79%
JPM(D) JPMorgan Chase & Co 224.71 2.50 1.13%
KO(D) Coca-Cola Co 72.49 0.44 0.61%
MCD(D) McDonald's Corp 288.49 0.63 0.22%
MMM(D) 3M Co 134.69 1.78 1.34%
Page 1
Untitled Workspace: 1
Buying Power $4,000,000 Open P/L $0 Closed P/L $0 03 Sep 2024 09:33 30 AM Sim Data

Figure 1.25 - Desktops

Desktops can be managed by going to **File** | **Desktops**.

In conclusion, TradeStation provides a comprehensive suite of tools and features for traders and investors, facilitating technical analysis and decision-making. The platform's tools, such as RadarScreen and Scanner, empower users to monitor and filter financial instruments based on specific criteria, while Chart Analysis allows for a visual representation of historical price movements. Traders can customize their analyses through various chart types, time frames, and technical indicators.

Summary

In this chapter, we have clarified what we mean by the term algorithmic trading and presented the tools we will be using. We have also introduced the working environment in which the author has developed his expertise. Then, we introduced the TradeStation platform, describing the download/installation process and the main platform functionalities. Being familiar with the main TradeStation functionalities will be a great help when we get our hands on EasyLanguage in the next chapter.

2

Getting Hands-On with EasyLanguage

The TradeStation website, and the web in general, offers a plethora of training material on EasyLanguage, which can be disorienting for those who want to start learning this language. In this chapter, we have organized the training material to provide you with an effective starting point, aiming to get you quickly writing your first scripts. The chapter requires familiarity with basic concepts of mathematics and financial markets, but no experience in other programming languages is necessary.

In this chapter, you will learn how to create an EasyLanguage script, write basic code, and apply scripts to charts. The goal of the chapter is to equip you with basic programming skills to succeed in the subsequent chapters.

This is what you will learn about in this chapter:

- Understanding the basics of EasyLanguage
- Writing indicators
- Writing strategies

Understanding the basics of EasyLanguage

This section explores the fundamentals of EasyLanguage, offering a comprehensive understanding of its syntax, structure, and capabilities. In this section, we will cover the following topics:

- What is EasyLanguage?
- How EasyLanguage works
- Words
- Expressions
- Statements
- Punctuation

What is EasyLanguage?

EasyLanguage comprises words, operators, and punctuation, amalgamated to formulate rules or instructions grounded in market data. These directives guide TradeStation in executing various analytical tasks. The rules and instructions, along with their resultant tasks, operate on the basis of “bars” and the data encapsulated within each bar.

The following primary data is contained in each bar of a chart:

- **Open**: The price level the bar opened at
- **High**: The highest price level in the bar
- **Low**: The lowest price level in the bar
- **Close**: The price the bar closed at
- **Date**: The date of the bar
- **Time**: The time of the bar
- **Volume**: The volumes traded in the bar

In *Figure 2.1*, you can see how EasyLanguage reads chart information:

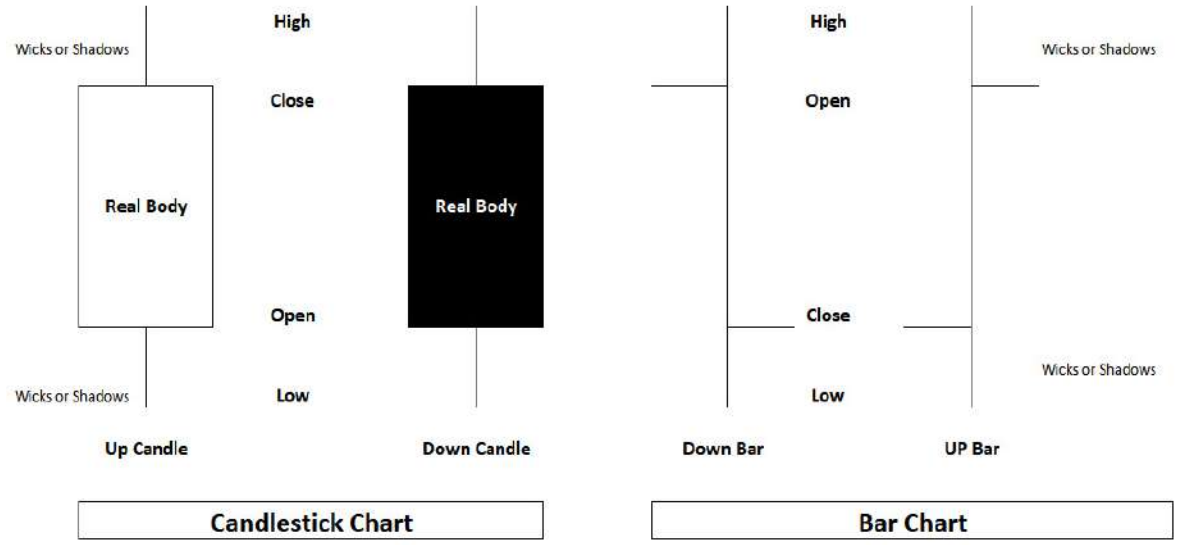


Figure 2.1 - Data contained in a candle (or bar)

We can see two types of charts in the preceding diagram: a candlestick chart and a bar chart. Candlestick charts are a type of financial chart used to represent the price movements of an asset over a specific period of time. Each candlestick typically consists of a rectangular body and two thin lines, called wicks or shadows, extending from it. The body of the candlestick represents the opening and closing prices of the asset during the given period, while the wicks indicate the highest and lowest prices reached.

Bars charts display the same information as candlestick charts, differing only in terms of their visualization. A bar chart consists of several components:

- The vertical line represents the price range for the given time period, with the bottom of the line indicating the lowest price reached and the top indicating the highest price
- A small horizontal line extending to the left from the vertical line represents the opening price of the asset

- Another small horizontal line extending to the right from the vertical line represents the closing price of the asset

> ***NOTE***
>
> *For algorithmic trading purposes, there is no difference between using bar charts and candlesticks charts; we will use the terms bar and candlestick (or candle) synonymously.*

How EasyLanguage works

TradeStation follows the EasyLanguage instructions for each bar on the chart line by line, from the top to the bottom of the EasyLanguage code, working across the chart bar by bar from left to right.

To see this in action with an example, let's assume we want to apply an indicator to a three-bar chart (*Figure 2.2*). Once you apply the indicator to the chart, TradeStation will perform the following tasks:

1. Load the data contained in **Bar1**.
2. Execute the script from top to bottom on the loaded data (it only has one bar of data in memory).
3. Load the data contained in **Bar2**.
4. Execute the script from top to bottom on the loaded data (it now has two bars of data in memory).
5. Load the data contained in **Bar3**.
6. Execute the script from top to bottom on the loaded data (it now has three bars of data in memory).

Figure 2.2 illustrates the preceding routine with TradeStation executing the indicator's script when applied to a chart:

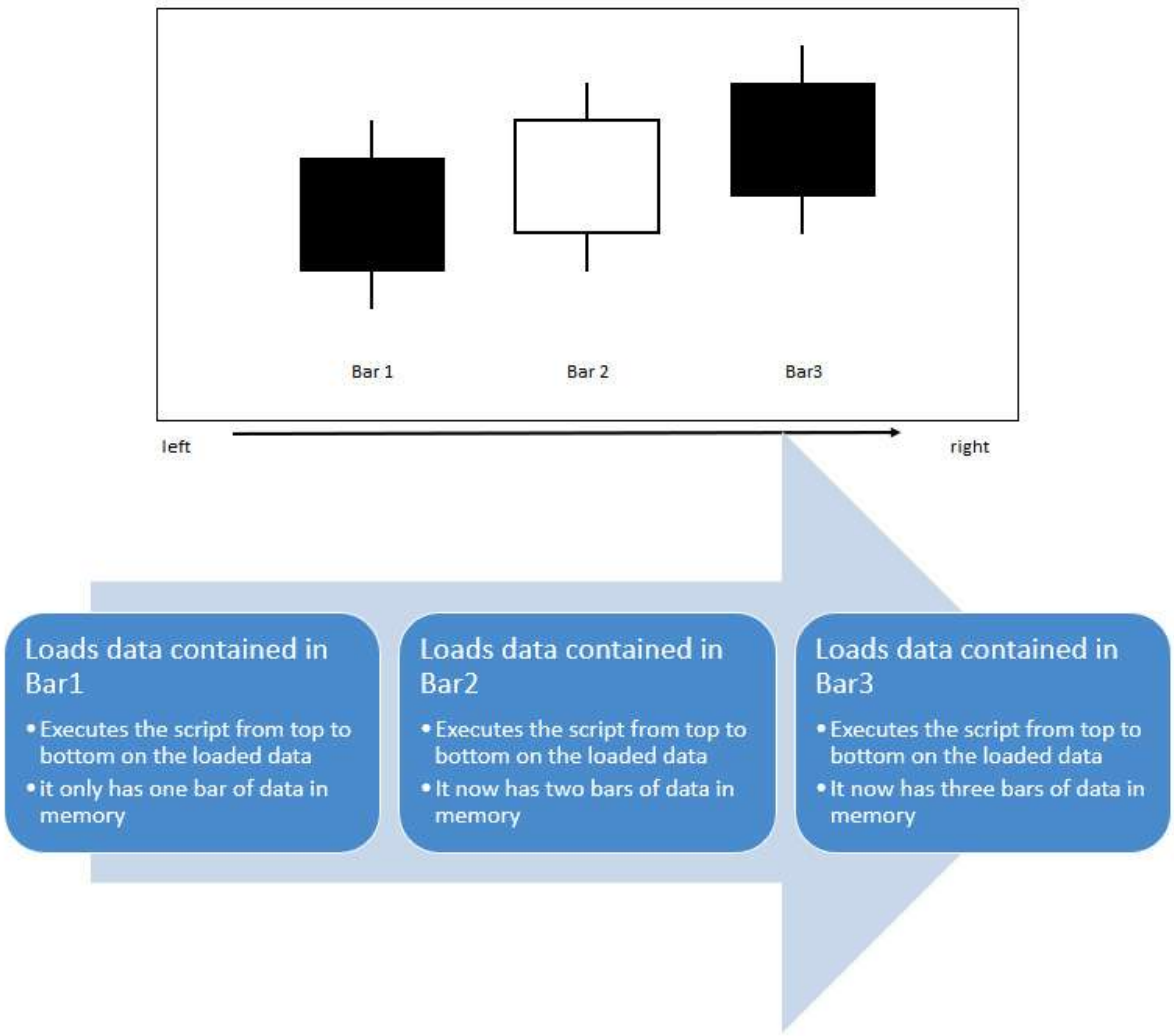

Bar 1
Bar 2
Bar3
left
right
Loads data contained in Bar1
• Executes the script from top to bottom on the loaded data
• it only has one bar of data in memory
Loads data contained in Bar2
• Executes the script from top to bottom on the loaded data
• It now has two bars of data in memory
Loads data contained in Bar3
• Executes the script from top to bottom on the loaded data
• It now has three bars of data in memory

Figure 2.2 - How EasyLanguage works

Looking at *Figure 2.2*, you can see that TradeStation loads one bar of data at a time and executes the EasyLanguage script each time for each new bar. In other words, it reads historical data sequentially, starting from the oldest available data point and moving forward in time.

The primary rationale behind EasyLanguage's operational method is to replicate traders' decision-making processes, a process that relies solely on historical data. Essentially, when scrutinizing any bar data, the program can only retrieve past data situated to the left of that specific bar on the chart. This constraint is vital to prevent the computation of future data, thereby emulating traders' reliance solely on known historical data.

In contrast, certain other programs opt to gather all historical data back to the first bar on the chart. This methodology may yield deceptive outcomes during the back-testing phase if not managed adeptly by experienced programmers.

EasyLanguage words

Words are the building blocks of EasyLanguage just as they are in any language. EasyLanguage is *not* case sensitive. Line returns, line spacing, and paragraph indents are for readability only. They do not affect the running of EasyLanguage instructions.

EasyLanguage words may be categorized generally as follows:

Type of word	**Descriptions**	**Examples**
Reserved words	Words with predefined meanings in EasyLanguage	**`Close`**, **`Open`**, **`Volume`**, **`High`**, **`Low`**, **`Input`**, **`Var`**, **`PlotN`**, **`Red`**, **`Green`**, **`Buy`**, **`Sell`**
Functions	Words that call a function (mainly stored technical indicators) in an EasyLanguage document	**`Average`**, **`RSI`**, **`Lowest`**, **`BollingerBands`**
User-defined words	Words users can create in their own analysis techniques	**`Input names`**, **`variable names`**
User-defined words containing text	Words users can create in their own analysis techniques containing text strings	**`Plot names`**
Skip words	Words used to improve readability but are skipped over by TradeStation when executing EasyLanguage instructions	**`of`**, **`the`**, **`at`**, **`from`**

Table 2.1 - EasyLanguage words

A great idea is to use syntax coloring, as it can really help in the coding process. For example, reserved words could be colored blue, functions could be colored magenta, and so on. You can enable this feature or change its settings from the Editor's menu by going to **Tools** | **Options** | **Syntax coloring**.

EasyLanguage expressions

One or more EasyLanguage words are used to form expressions, which are essentially like phrases in any language. Two of the most common types of expressions are the following:

- **Numeric expressions**, which refer to or calculate a value:
    - If we wanted to multiply the volume by 2, we could write **`Volume * 2`**
    - If we wanted to divide the range of the bar by 2, we could write **`(High – Low) /2`**
    - An expression could also be made up of just one reserved word, for example, **`Close`**
- **True/false expressions**, which compare two numeric expressions (values) and are either true or false:
    - If we need a Boolean instruction that returns **`True`** if the close is equal to the open, we could use the following expression: **`Close = Open`**
    - If we need a Boolean instruction that returns **`True`** if a given bar's high is greater than the previous bar's high, we could write the following expression: **`High > High[1]`**
    - If we need a Boolean instruction that returns **`True`** if the date has changed from the previous bar, we could use the following expression: **`Date <> Date[1]`**

EasyLanguage statements

One or more expressions are put together to form a statement, which is like a sentence. All EasyLanguage instructions consist of statements. Every statement ends with a semicolon.

A few of the most common statements are the following:

- Plot statements:

  ```
  Plot1( Close, "The Close");
  ```

- Declaration statements:

  ```
  Inputs: Length(10);
  ```

- Variables:

  ```
  NetChange(0);
  ```

- Variable assignment statements:

  ```
  Value1 = Close – Close[1];
  ```

- If/then (conditional) statements:

  ```
  If Range > Average(Range, 5) then Alert;
  ```

EasyLanguage punctuation

EasyLanguage requires the proper use of punctuation. The following table summarizes the most common EasyLanguage punctuation marks and their most common uses:

	Punctuation	Description

	Punctuation	Description
( )	Parentheses	Used in statement structures, also for grouping mathematical operations
“ “	Quotes	Used to denote a text item (e.g., plot name) within EasyLanguage
;	Semicolon	Indicates the end of a statement
:	Colon	Denotes the declaration of a list
,	Comma	Separates items in a list
[ ]	Square brackets	Used to reference data from a previous bar and to displace a plot; also used in arrays
{ }	Curly brackets	Any text between the brackets is treated as a comment (used in coding for remarks or notation) and not as part of the EasyLanguage instructions
//	Double slash	Text after the double slash on a line is considered a comment and is not considered part of the EasyLanguage instruction for the rest of that line

Table 2.2 - EasyLanguage punctuation

As a general guideline, TradeStation employs the terms *indicators*, *analysis techniques*, and *studies* interchangeably to describe the tools to visualize information on a chart. We are going to learn how to program such tools in the *Writing indicators* section. By contrast, TradeStation uses the terms *strategy* and *signal* to denote rule-based trading tools. We are going to learn how to program these tools in the later *Writing strategies* section.

Before we move on, just a note about menus. In TradeStation, there are several menus:

- We will use the term *Main menu* to refer to the main platform menu
- We will use the term *Editor’s menu* to refer to the TradeStation Development Environment menu
- To refer to the menu for a specific application, we will use the name of the application + menu (for example, Chart’s menu, RadarScreen’s menu, and so on)

We are now familiar with EasyLanguage grammar, so let’s start coding.

Writing indicators

Many trading projects and ideas arise from observing charts; this is the reason why indicators are at the basis of algorithmic trading. Indicators consist of mathematical calculations made on data that are used to draw plots on a chart. These plots are usually lines, histograms, or dots. Please note that TradeStation often uses the terms *indicators*, *studies*, and *analysis techniques* synonymously.

We are now going to program the following indicators:

Exercise	**Indicators**	**Learning Purpose**
01	Close	Using one plot statement in an indicator
02	Close Open	Using two plots statements in an indicator
03	RealBody	Using mathematical operations in a numeric expression in a plot statement
04	MidPrice	Using parentheses in numeric expressions to order operations correctly
05	NetChange	Referencing data from previous bars
06	NetChange with variables	Using user-declared numeric variables—writing variable declaration statements and variable assignment statements
07	Momentum	Using functions
08	MovingAverages	Working with inputs
09	Crossover	Writing a ShowMe study; using block if/then statements
10	Uptrend	Writing paintbars; using block if/then/else statements
11	Time	Referencing time using the AND logical operator

Table 2.3 - Exercises list

Before starting our exercises, please ensure you are familiar with the following most common commands, as we will use them often in this chapter:

Purpose	**Command**
To open the EasyLanguage Editor	From the **Main** menu, go to **File** | **New_Application** | **EasyLanguage**
To create a new indicator	From the Editor's menu, go to **File** | **New** | **Indicator**
To compile scripts	From the Editor's menu, go to **File** | **Build** | **Verify** (or press *F3*)
To create a new chart	From the **Main** menu, go to **File** | **New_Application** | **Chart Analysis** (or press *Ctrl + Alt + C*)
To apply an indicator to a chart	From Chart's menu, go to **Studies** | **Add_Study**
To save the workspace	From the **Main** menu, go to **File** | **Workspace** | **Save Workspace_As**

Table 2.4 - Most common commands

Let's begin!

Exercise 01—Close

Let's see how to use a plot statement in our indicators:

```
  Plot1(Close,"MyClose");
```

`Plot1` is a reserved word used to plot values on a price chart.

The plot syntax is the following:

```
PlotN(Expression,"<PlotName>");
```

Here, **`N`** is a number between 1 and 99, representing one of the 99 available plots. **`Expression`** is the value to be plotted and **`<PlotName>`** is the name of the plot. In the following figure, you can see how the chart will look after applying the indicator on it:

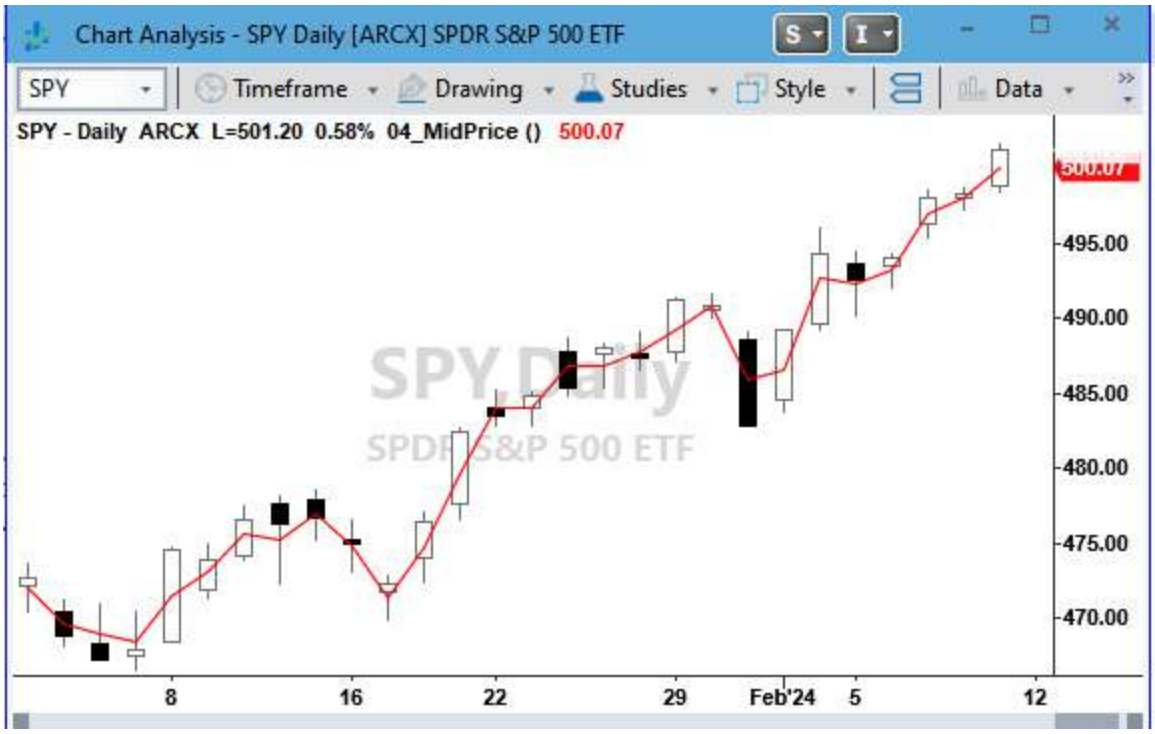


Figure 2.3 - Plotting a line connecting the close of each bar

Exercise 02—Close Open

Let's now move on to see how to use two plot statements in our indicators:

```
Plot1(Open,"Open");
Plot2(Close,"Close");
```

This indicator will plot one line connecting the close of each bar and a second line connecting the open of each bar. It requires two plot statements; remember that all statements require a semicolon (`;`) at the end.

In *Figure 2.4*, you can see the indicator applied to the chart:

Figure 2.4 - Connecting the close and open prices of each bar

Exercise 03—Real Body

Let’s now have a look at how to use mathematical operations in a plot statement:

```
Plot1(Close - Open,"Real Body");
```

In candlestick charting, the “real body” refers to the difference between the close and the open of each bar; if the close is greater than the open, it will be a positive number, while if the close is lower than the open, it will be a negative number. Given the different scale to that used with prices, this plot needs to be shown in a sub-chart separate from the prices one.

In order to assign a sub-chart, go to **Studies** | **Edit Study** | **Customize** | **Scaling** and choose the preferred sub-graph. In this example, we chose sub-graph 2.

TradeStation calculates arithmetic expressions as follows:

- Multiplication and division from left to right, followed by addition and subtraction from left to right.
- This order may be customized through the use of parentheses. Operations inside parentheses are done first, starting with the innermost parentheses.

  To clarify, consider the following examples:

  1+5*3+1=17

  (1+5)*(3+4)=42

  1+(5*(3+4))=36

In *Figure 2.5*, you can see the Real Body indicator applied to a sub-graph.

Figure 2.5 - The Real Body indicator plotted in sub-graph 2

As you see in *Figure 2.5*, we plotted the Real Body indicator as a histogram. To change the indicator's style, go to Chart's menu and select **Studies** | **Edit Studies** | **Customize** | **Style**.

Exercise 04—MidPrice

Let's see how to use parentheses in numeric expressions to order operations correctly by calculating the mid price. The mid price is the price level you obtain by summing the high and the low of each bar and then dividing the result by 2:

```
Plot1( (High + Low ) / 2 ,"MidPrice");
```

The `(High + Low)/2` instruction calculates the mid price correctly.

If you use the parentheses in the wrong way, the script might be formally correct, but the formula calculation can be wrong. For example, say you write the following:

```
Plot1( High + Low  / 2 ,"MidPrice");
```

The indicator will compile, but the plot it makes will be incorrect, because the expression `High+Low/2` is not the mid price, which is `(High+Low)/2`.

In *Figure 2.6*, you can see the MidPrice indicator applied to a graph.

Figure 2.6 - A line connecting the mid price values on our chart

Looking at the chart in *Figure 2.6*, we can see the Mid Price line is exactly at the middle of each bar.

Exercise 05—NetCh

Referencing data from previous bars is crucial when working with historical data. Let's assume you want to plot the difference between the current close and the close of five bars prior. You would use the following code:

```
plot1( close - close[5],"NetCh");
```

In order to reference data from previous bars, you need to use square brackets, **`[ ]`**. In this example, **`close[5]`** refers to the close of five bars prior. Remember that by *current bar*, we mean the bar that TradeStation is currently reading (remember what we learned in *Chapter 1*). We can refer to the current bar using **`[0]`**, to the bar at the left of the current bar with **`[1]`**, and so on in a leftward direction across the chart.

The chart showing the net change indicator looks as follows:

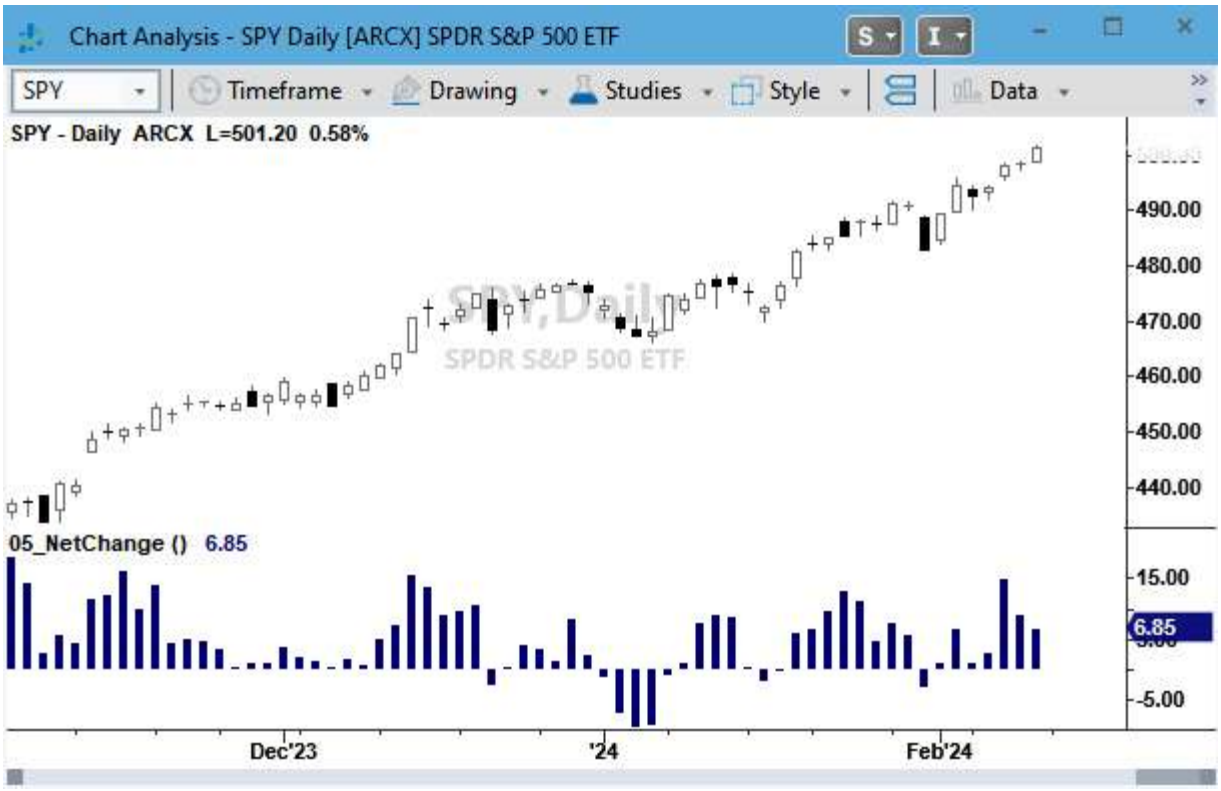


Figure 2.7 - Net change versus the close from five bars ago

Exercise 06—NetCh with variables

Our next learning target concerns how to use user-declared numeric variables, variable declaration statements, and variable assignment statements.

Let's take the script from *Exercise 05* and rewrite it as follows:

```
Var:NetCh(0);
NetCh = close - close[5];
plot1(NetCh,"NetCh");
```

As you can see, we made the following amendments:

1. We added the **`Var:NetCh(0);`** statement to declare the numeric variable named **`NetCh`**. TradeStation understands this is a numeric variable because it reads a number between parentheses.
2. Then we assigned the variable the following formula: **`NetCh = close - close[5];`**.
3. Then we put the **`NetCh`** variable inside the plot statement by replacing **`close - close[5]`** with **`NetCh`**.

In programming, a variable is a symbolic name or identifier used to store data temporarily in a computer's memory. When you create a variable in EasyLanguage, you need to specify its name and the type of data it will hold. Once defined, you can assign values to the variable, update its value, or perform operations using its value within your program.

> ***NOTE***
>
> *Each variable is only valid within the given analysis technique or strategy in which it is declared and/or assigned.*

Exercise 07—Momentum

Let's now move on to the use of functions. Let's start with the built-in **`Momentum`** function:

```
plot1(Momentum(close,10),"Momentum");
```

Functions are frequently used formulas (or comparisons) that return numeric values (or **`true`** or **`false`**). They can be called in any analysis technique or strategy with just a few words, eliminating the need to re-create complex formulas. Functions can be used in any set of EasyLanguage rules or instructions.

In the preceding code, we are using the built-in **`Momentum`** function, which requires two parameters: the price (we used close) and the length (we used **`10`**).

TradeStation stores the most common mathematical and statistical formulas (such as average and standard deviation) and the most common technical analysis formulas (such as RSI and ADX) as built-in functions. In order to access the built-in functions library from the Editor's menu, go to **Help | EasyLanguage Reserved Words and Function**.

In *Figure 2.8*, you can see the Momentum indicator applied to a chart:

Figure 2.8 - Momentum function

As you see in *Figure 2.8*, the Momentum indicator is also displayed in a sub-chart because it needs a different scale from prices.

Exercise 08—Moving Average

Let's use another built-in function, the average, to learn how to work with inputs:

```
Input:Length(200);
var: SMA(0);
SMA =average(close,Length);
plot1(SMA,"SMA");
```

Moving averages are a statistical technique used to analyze data points by creating a series of averages of different subsets of the full dataset. The idea is to smooth out short-term fluctuations and highlight longer-term trends or patterns in the data. There are different types of moving averages; in this example, we focus on the **`average`** function, which calculates the **Simple Moving Average** (**SMA**).

`Input` is a user-editable value used in an analysis technique or strategy. Each **`Input`** carries a user-declared name and default value. Inputs are declared using an **`Input`** declaration statement, such as **`Input:Length(200);`**, where the value of **`200`** creates a 200-day SMA indicator.

In *Figure 2.9*, we can see what the SMA indicator looks like:

Figure 2.9 - SMA indicator with an input length of 200

In *Figure 2.9*, you can see an SMA plotted with an input length of **`200`**. To edit the input values from Chart's menu, go to **Studies** | **Edit Studies** | **Customize** | **Input**, then change the **Length** value to meet your requirements.

> ***NOTE***
>
> *Each input is only valid within the analysis technique or strategy in which it is declared.*

Exercise 09—Crossover

We are now going to learn how to work with ShowMe plots. A ShowMe serves as a TradeStation tool designed to spotlight only select bars that fulfill particular criteria on a given chart. In order to create a new ShowMe, go to the Editor's menu and select **File** | **New** | **ShowMe**.

In this script, you are going to learn how to use the **`if/then`** block:

```
If close cross over average(close,20)
then plot1(Close,"Crossover");
```

This ShowMe marks only those bars crossing the 20-day SMA.

`If close cross over average(close,20)` returns **`True`** or **`False`**. If the expression is **`true`**, the ShowMe plots a dot at the close of the bar; if it returns **`false`**, it doesn't plot anything.

In *Figure 2.10*, you can see how a ShowMe plot appears on a chart:

Figure 2.10 - A ShowMe plot highlighting closes that cross over the SMA

As you can see in *Figure 2.10*, the indicator plots a dot any time the close of the bar crosses over the 20-day SMA.

Exercise 10—Uptrend

We are now going to learn how to work with PaintBars. A PaintBar is a TradeStation functionality that lets us highlight groups of bars that meet some criteria using different colors. In order to create a new PaintBar, go to the Editor's menu and select **File** | **New** | **PaintBar**.

This indicator will color only bars with a close above the 200-day SMA:

```
if close > average(close,200) then
    plotPB(High,Low,"Uptrend",darkgreen)
else
   plotPB(High,Low,"Uptrend",red);
```

The **`plotPB`** reserved word is an abbreviated version of **`PlotPaintBar`** and enables you to paint the entire bar a specified color or even paint a section of the bar between two specified values. PaintBars require at least two arguments. In this example, we used **`High, Low`**.

PaintBars are highly beneficial in accentuating chart setups through color without adding extra lines, thus maintaining the chart's clarity. Please note that the indentation in the preceding code is only for clarity purposes.

In *Figure 2.11*, you can see how the Uptrend PaintBar looks when plotted on a chart.

Figure 2.11 - Uptrend PaintBar

In *Figure 2.11*, observe the functionality of the **`if/then/else`** block, plotting bars in dark green if their close is above the 200-day SMA, and in red otherwise.

Exercise 11—Time

Let's now write a new PaintBar to color the bars according to a time reference:

```
if time>=0930 AND time<=1130
then plotPB(High,Low,"Time");
```

In this PaintBar exercise, we are going to learn the following:

- How to reference time
- How to use the **`AND`** logical operator

How to reference time

Every bar in a **Chart Analysis** window has an ending timestamp. The time is referenced according to military format with no colon:

- Time = 900 (or Time=0900) references a bar with an ending time of 9:00 A.M.
- Time = 1800 references a bar with an ending time of 6:00 P.M.

How to use the AND logical operator

In this exercise's example, the **`PlotPB`** command depends on whether **`time>=0930 AND time<=1130`** evaluates to true or false, which is **`true`** if the bar's timestamp is between 9:30 A.M. and 11:30 A.M; other times outside of this range will be ignored.

The **`AND`** operator ensures that both the **`time>=0930`** and **`time<=1130`** expressions are true.

In *Figure 2.12*, you can see how the PaintBar appears on an intraday chart:

Figure 2.12 - Time-based PaintBar

As you can see in *Figure 2.12*, we plotted a 15-minute chart where all the bars with a timestamp between 9:30 A.M. and 11:30 A.M. are colored red. This way of using PaintBars on intraday charts is very useful for algorithmic day trading. In TradeStation, indicators, ShowMe plots, and PaintBars are the basis for algorithmic trading for three main reasons:

- In my experience, many algorithmic trading ideas are born from chart visualizations
- Coding indicators provide the starting point for coding strategies
- Plotting indicators on the same chart as strategies will help a lot in correcting logical errors

We are now familiar with the scripts used for indicators, so we can move on to strategies.

Writing strategies

Let's now move from indicators to strategies. A TradeStation strategy is a tool to automate rule-based trading signals, back-test them on historical data, and use them in real trades. Please bear in mind that in TradeStation, all strategies need charts on which they can be applied, and strategies use the same chart's historical and real-time data.

In this section, we are going to learn about the following:

- How to program a basic strategy
- The order syntax
- MaxBarsBack

- How to program **take profit** and **stop loss** levels

How to program a basic strategy

In order to create a new strategy, follow these instructions:

1. From the Editor's menu, go to **File** | **New** | **Strategy**.
2. Type the name you want to assign.
3. Click **OK**.
4. If the EasyLanguage Editor displays a default script, delete it by pressing *Ctrl + A* and then *Ctrl + Z*.

Let's write a basic strategy to get to grips with order syntax:

```
Input: Length(200);
if close crosses above average(close, Length) then buy next bar open;
if close crosses below average(close, Length) then sell  next bar open;
```

In the preceding script, we have three statements:

- The first statement, **`Input: Length(200);`**, declares an input for the length of the moving average, which we set to 200 days
- In the second statement, we have two expressions:
    - **`if close crosses above average(close, Length)`** sets the condition upon which to buy
    - **`buy next bar open`** is the instruction to buy
- In the third statement, we also have two expressions:
    - **`if close crosses below average(close, Length)`** sets the condition upon which to sell
    - **`sell next bar open`** is the instruction to sell

Let's now dive into EasyLanguage order syntax.

EasyLanguage order syntax

EasyLanguage order syntax is made of two components:

- The market action
- The order type

EasyLanguage uses four verbs to identify the market action to be taken, as outlined in the following table:

Market Action	**Description**

Market Action	Description
Buy	Establish, or add to, a long position (any existing short position will be covered)
SellShort	Establish, or add to, a short position (any existing long position will be liquidated before a short position is established)
Sell	Sell to liquidate a long position only
BuyToCover	Buy to cover a short position only

Table 2.5 - Market actions

These actions may be taken according to different order types, as detailed in the following table:

Order Type	Description	Market Rationale
`Next bar at market`	The order will take place on the open of the next bar	If you are in a hurry to buy or sell, you can use market orders. Such orders depend on the broker and might be executed at a worse price than expected. This issue will be taken into consideration in the coming chapters, when we analyze strategies for back-testing results.
`Limit`	The order will take place on the next bar if the limit price is reached	This could be used when you want to buy at a lower price than the current close, or you want to sell at a higher price than the current close.
`Stop`	The order will take place on the next bar if the stop is triggered	You could use this when you want to buy when the price breaks up a certain level above the current close, or you want to sell when the price breaks through a certain level below the current close.

Order Type	Description	Market Rationale
`This bar on close`	The order will take place on the close of this bar (back-testing only)	This instruction will send a market order on the next bar's open in real time.

Table 2.6 - Order types

In *Figure 2.13*, you can see how the chart looks after applying a strategy to it:

Figure 2.13 - MovingAverage strategy and indicator

If you observe *Figure 2.13* closely, you will notice the following:

- Any time the close crosses above the 200-day SMA, in the next bar, we see the following:
    - A blue up arrow appears to highlight a new open position
    - A blue triangle marks the entry price at the open of the bar
- Any time the close crosses below the 200-day SMA, in the next bar, we see the following:
    - A red down arrow appears to highlight the existing position that has been closed
    - A red triangle marks the exit price at the open of the bar
- **100** indicates how many shares have been traded
- A cyan line connects the entry and exit bars in winning trades

- A red line connects the entry and exit prices in losing trades

MaxBarsBack

A strategy requires a minimum number of bars to calculate the first value. For example, the previous strategy needs at least 200 bars to calculate the 200-day SMA. You need to provide this information, which TradeStation refers to with the term **MaxBarsBack**.

If you get an error saying **Tried to reference back more bars than allowed by the current MaxBarsBacking setting. Please increase the MaxBarsBack setting**, follow these steps:

1. From Chart's menu, go to **Studies** | **Edit Strategies** | **Properties for all**.
2. Then change the **Maximum number of bars study will reference** value to **`200`**, as shown in the following screenshot.

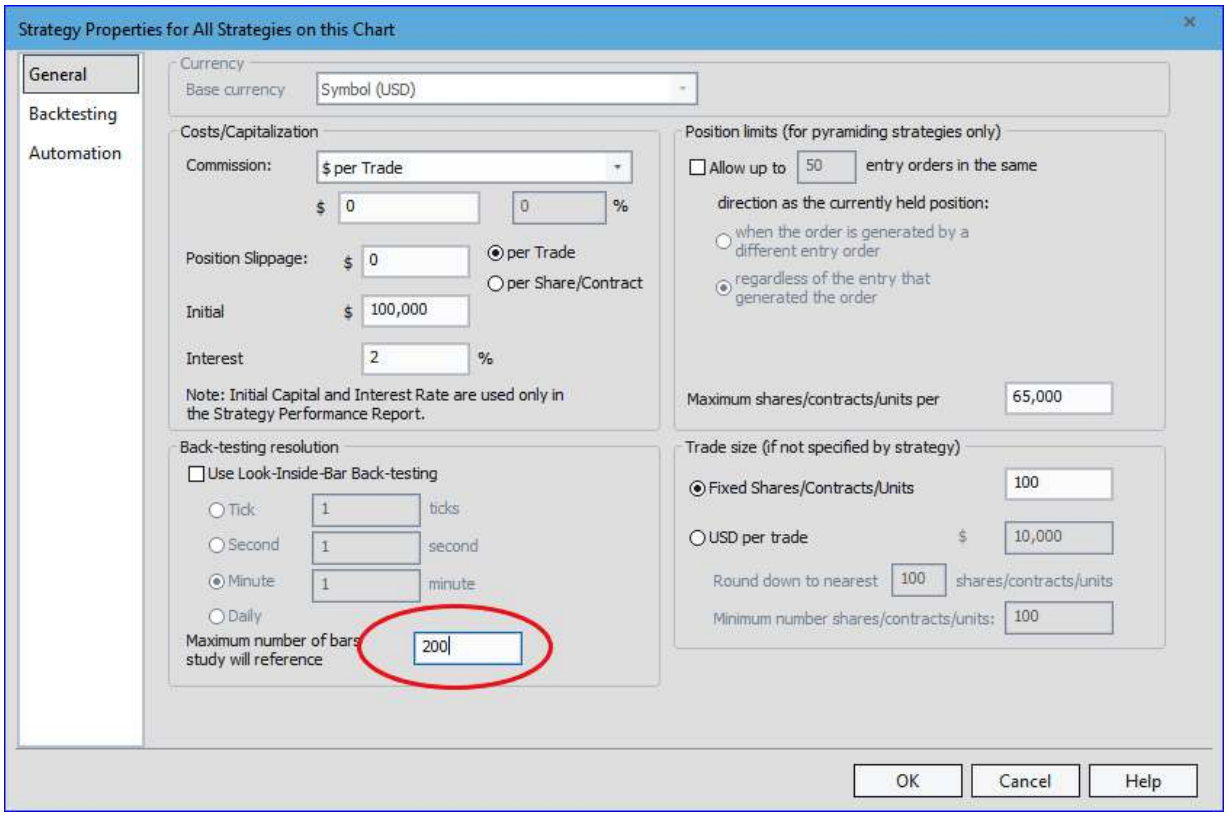


Figure 2.14 - MaxBarsBack

In *Figure 2.14*, you can see how to use the **Strategy Properties** window to change the MaxBarsBack value.

Please note that the MaxBarsBack size reduces the historical data available for back-testing. In the previous example, the first 200 days of historical data will be ignored in the back-testing process.

How to program take profit and stop loss levels

A **take profit** order is an instruction given to a broker or trading platform to automatically sell an asset when its price reaches a predetermined level that would result in a profit for the trader. This level is often set above the current market price for long positions (buy orders) and below the current market price for short positions (sell orders). Take profit orders help traders lock in profits and avoid the risk of losing gains if the market reverses direction after a favorable price movement.

A **stop loss** order is an instruction given to a broker or trading platform to automatically sell an asset when its price reaches a predetermined level that would result in a loss for the trader. This level is often set below the current market price for long positions and above the current market price for short positions. Stop loss orders help traders limit potential losses by closing out losing positions before they can accumulate further losses.

Both take profit and stop loss orders are essential risk management tools used by traders to protect their capital and manage their investment portfolios effectively. By setting these orders, traders can define their risk-reward ratios and establish clear exit points for their trades, which helps them maintain discipline and control emotions during trading.

In this example, we are going to set a 50-cent take profit order and a 10-cent stop loss order for any long position open on SPY:

```
Input:Length(200);
if close crosses above average(close,Length) then buy  next bar open;
if close crosses below average(close,Length) then sell  next bar open;
if Marketposition = 1 then begin
    sell ("target") next bar entryprice + 0.50 limit;
    sell ("stoploss") next bar entryprice – 0.10 stop;
```

`Marketposition` is a reserved word to refer to the strategy position and can assume one of three values:

- **`Marketposition = 0`** means that there is no open position
- **`Marketposition = 1`** means there is a long position opened
- **`Marketposition = -1`** means there is a short position opened

`entryprice` is a reserved word that provides the entry price for the current position.

The preceding example script performs two tasks:

- Take a profit exiting the long position if the price, in any bars following the entry bar, touches the entry price plus 50 cents
- Take a stop loss exiting the long position if the price, in any bars following the entry bar, touches the entry price minus 10 cents

Figure 2.15 - Stop loss and take profit

Please observe *Figure 2.15*, where the red arrow at the top right of the chart marks the target exit.

If you hold the mouse exactly on the exit bar, you can read the exit price of **484.01**, which is exactly 50 cents above the entry price. Remember that when using limit orders to exit positions, the trade can occur any time inside the bar without waiting for the close.

The same concept applies to stop orders. In *Figure 2.16*, we can see that the strategy exited a long position with a stop loss at 256.17, which is exactly 10 cents below the entry price.

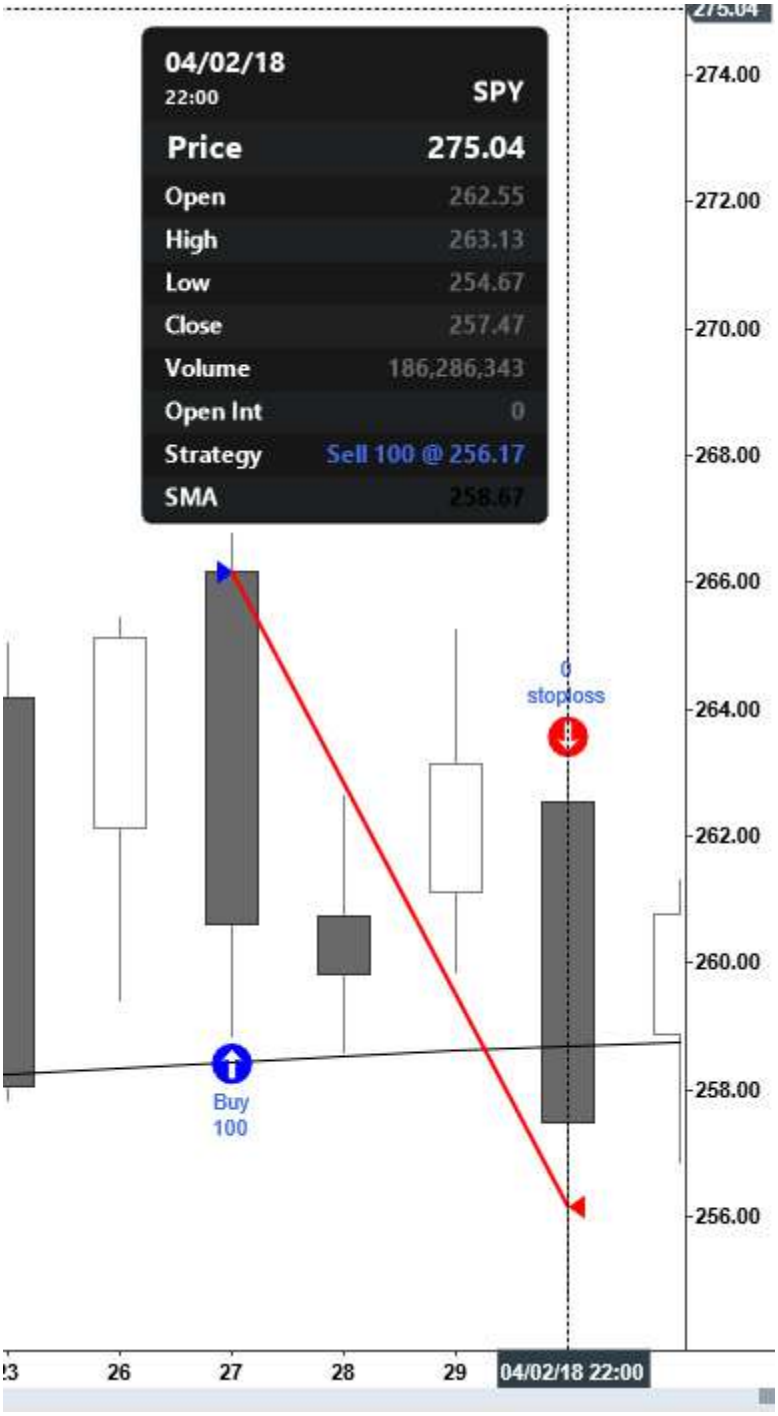


Figure 2.16 - Stop loss level

IMPORTANT NOTE

Remember that you should be very careful when using stop orders in real markets, as, depending on the market volatility and volumes, the order might be executed far from the price that was read at the moment you sent the order.

In this section, we learned how to write EasyLanguage strategies, and we saw how in TradeStation, unlike other programming languages such as Python, each strategy requires a chart to be applied.

Summary

In this chapter, we learned the basics of EasyLanguage, how to manage the TradeStation development environment, how to program simple indicators and strategies, and how to handle both of them together as a whole. What you learned in this chapter will be very useful in the next chapters when you build real-life trading tools. This, of course, doesn't provide exhaustive coverage of the vast subject matter related to EasyLanguage. On the TradeStation website, you can easily find a wealth of useful material to expand your knowledge and delve into specific topics.

In any case, if you have made it this far, it means you have acquired the basics necessary to progress to the next chapter, where, while the focus will still remain on learning EasyLanguage, we are going to change the perspective a bit by dealing with real cases encountered in the professional world. This is done with the purpose of preparing you for the real challenge that all algorithmic traders have to face, which is having a trading plan and sticking to it while navigating the market.

3

Writing a Trend Strategy

Now that we have gained confidence with basic indicators and strategy scripts, let’s move on to some real approaches that are used in a professional environment. In this chapter, we are going to learn what the market rationale behind a trend strategy is, how to deal with a trend algorithmically, and how to write trend strategies in EasyLanguage.

The end goal of this chapter is for you to understand the process of developing trading ideas and transforming them into algorithmic trading tools. In this chapter, we are going to deal mainly with equities, but the same concepts will apply to other markets, such as commodities and **Foreign Exchange** (**Forex**).

We are going to delve into the following topics:

- Developing trading ideas
- Market rationale behind a trend strategy
- Identifying a trend algorithmically
- Handling noise

Developing trading ideas

The process of algorithmic trading always starts with a trading idea and ends with the paper or live-trading activity; it is composed of the five main steps that you can see in *Figure 3.1*:

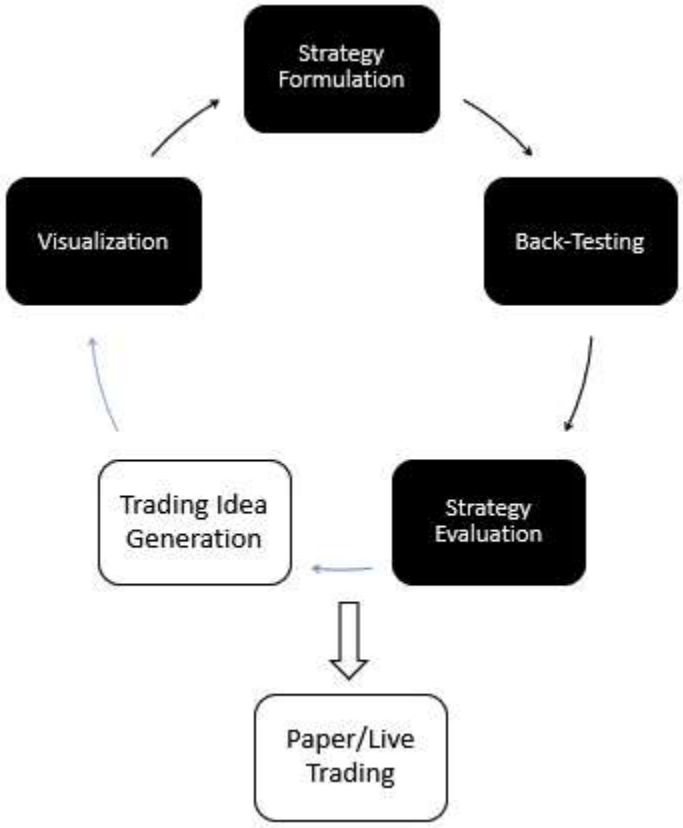


Figure 3.1 - Developing trading ideas

Looking at *Figure 3.1*, you can see how the process of transforming a trading idea into a trading algorithm works. Please keep this in mind while reading this book. Let's go through each step in detail:

1. **Trading idea generation**: This is the initial phase, where you come up with a trading strategy or idea. It could be based on technical analysis, fundamental analysis, quantitative analysis, machine learning, or a combination of these approaches.
2. **Visualization**: This step requires you to visualize the trading idea by creating all the necessary indicators and applying them to the financial instrument's price chart.
3. **Strategy formulation**: This step requires you to define the conditions for entering and exiting trades, risk management rules, position sizing, and so on. This step involves converting your idea into a set of rules or algorithms that a computer can understand.
4. **Back-testing**: This step is crucial to test the strategy's performance on historical data. Back-testing involves running your algorithm on past market data to simulate how it would have performed. This helps you evaluate its profitability, risk, and robustness.
5. **Strategy evaluation**: After back-testing, you need to analyze financial results. TradeStation offers a straightforward tool to achieve such a task, which is the Performance Summary report. This is a tool that provides a comprehensive overview of strategy performance metrics.

The process of developing a trading strategy is a journey that requires continuous adaptation. It's important to understand that it's not a linear path but rather an iterative process that may require multiple attempts and adjustments of different phases before achieving a satisfactory strategy.

For a trading idea to be effective, it must be explainable verbally, not just through numbers. This ensures a thorough understanding of the reasons behind trading decisions, avoiding excessive reliance on algorithms. A balanced approach combining quantitative analysis with qualitative market understanding is essential for adapting to changing market conditions. The ability to logically explain a trading idea in words underscores a solid and thorough market understanding. This type of understanding enables traders to make informed and lasting decisions, even when numerical data is limited or not clearly defined.

Let’s now discuss the rationale behind a trend strategy.

Market rationale behind a trend strategy

Market dynamics are primarily propelled by information, with market participants adjusting their positions in response to new data, thereby impacting the prices and volumes of financial assets, creating a continuous cycle of information-driven trading activity (*Figure 3.2*):

Figure 3.2 - An information feedback loop in market dynamics

Let’s delve into the market dynamics by analyzing the elements of *Figure 3.2*:

- Information

- Market participants
- Price/volumes

Information

The relevant financial information for algorithmic trading can be classified as in *Figure 3.3*.

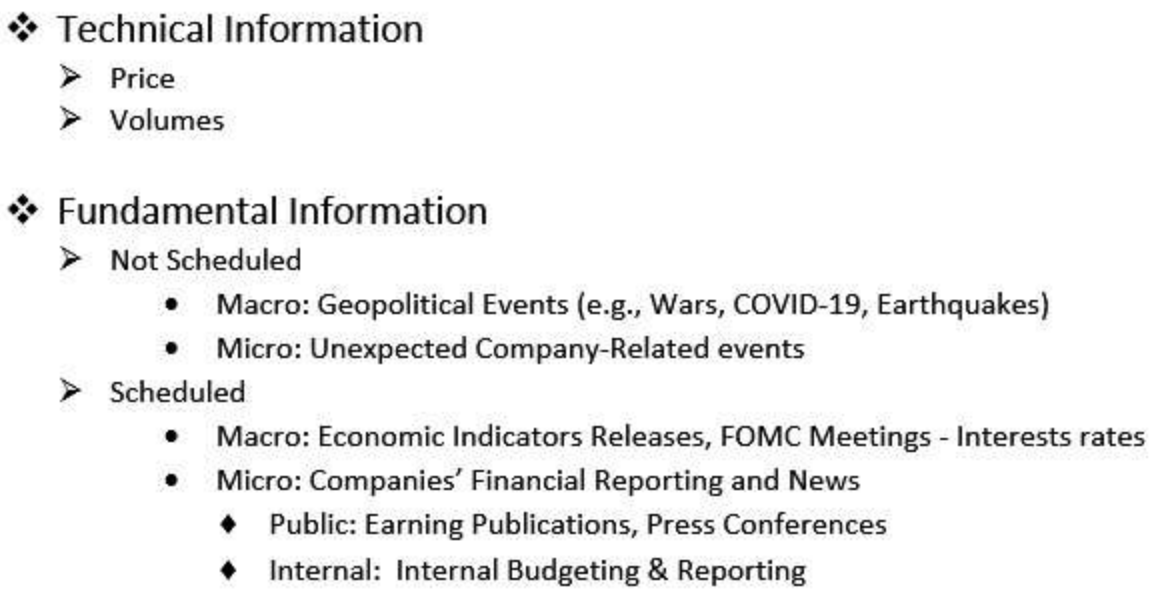


Figure 3.3 - Financial information classification

As you can see in *Figure 3.3*, the information that influences the financial market can be of a technical or fundamental nature. We will talk about **technical information** when it is directly related to the prices or volumes of financial instruments; for example, the value of a moving average is technical information. **Fundamental information** is a residual category; that is, we will consider fundamental information to be all information that is not derived from price and/or traded volumes able to influence market participants and price (*Figure 3.3*). Algorithmic traders will be more focused on analyzing technical information, but it is important for them to understand the functioning of the market from a fundamental perspective.

For example, some fundamental information is scheduled according to economic calendars; other information is not because it is completely unexpected. The unscheduled information is important for writing algorithms with a precise objective. For example, one can develop algorithms that intervene only in the event of sudden spikes in volatility caused by completely unexpected events. A typical example, as we will see in the chapter on risk management, is the volatility stop loss. Alternatively, one can create algorithms that follow the regularity of financial information releases during a trading session (in *Chapter 8*, we will see how to set up algorithmic strategies based on trading hours). In

turn, fundamental information can be classified into macro information and micro information depending on its impact, either only on the company being analyzed or on the entire sector or whole economy. Scheduled micro-type information can in turn be classified as public or internal.

As an algorithmic trader or programmer, you have probably learned that your focus is on technical information only, and this is true. However, in order to build a grounded strategy, you need to know how the fundamental information works with trends.

So, even if you are a technical algorithmic trader or a number-crunching programmer, please take a moment to carefully read the following lines regarding the specific impact of internal budgeting and reporting financial information on equity prices.

The internal budgeting and reporting cycle

In a corporate US company, there are two distinct working rhythms: operational and financial.

The **operational rhythm** is contingent upon the specific sector. For instance, in the ice cream business, the workload experiences a significant surge in the summer, while in the gas industry, the workload generally increases during the winter when temperatures decrease.

On the contrary, the **financial rhythm** remains more or less consistent across various businesses and sectors. This consistency is attributed to the budgeting and reporting process, which forms the foundation of the financial operations in companies.

If you enter the offices of any listed company in the last days of the month, or even more so in the last days of the quarter, you can feel the stress arising from financial deadlines. This stress will only subside after financial data has been collected, condensed, and sent to the central offices.

It’s important to note that listed companies generally have daily or weekly internal reporting on sales data, while costs are accounted for on a monthly basis. Every time fresh data arrives, even if only related to sales or only on a daily basis, the finance department conducts a **reforecast**, providing the company’s management with a revised forecast for the monthly, quarterly, and annual financial closing.

This means that fundamental information regarding the company’s financial statements *doesn’t suddenly come with the publication of quarterly results* but is built day by day. The company’s administration team has access to this information first, and, without questioning their integrity, it is highly unlikely that such information will remain a secret until the financial statements are published, which happens with a delay of several months. This is how the price trend emerges — from the gradual dissemination of fundamental information first among insiders and then among the general public.

Such concepts will be crucial when we set algorithmic tools' lengths. For example, we could use a 20-day moving average to try catching a trend starting from the monthly internal reporting cycle, a 50-day moving average to catch a trend based on a quarterly information cycle, or a 200-day moving average to identify a yearly trend based on financial year results.

Market participants

Market participants can be classified into two main categories based on their roles and activities within financial markets:

- **Retail investors** are individual investors who trade securities for their personal investment portfolios
- **Institutional investors** are large entities such as mutual funds, pension funds, hedge funds, and insurance companies that invest substantial amounts of capital on behalf of their clients or shareholders

Prices and volumes

Price is the value at which a financial instrument (such as a stock, bond, commodity, or currency) is bought or sold in the market. Prices can fluctuate due to various factors, including supply and demand dynamics, investor sentiment, economic indicators, geopolitical events, and company-specific news.

Volumes are the quantities of shares or contracts traded for a particular financial instrument within a given time frame. Volume reflects the level of market activity and liquidity for a security. Higher volumes generally indicate greater market interest and participation, while lower volumes may suggest reduced trading activity or lack of liquidity.

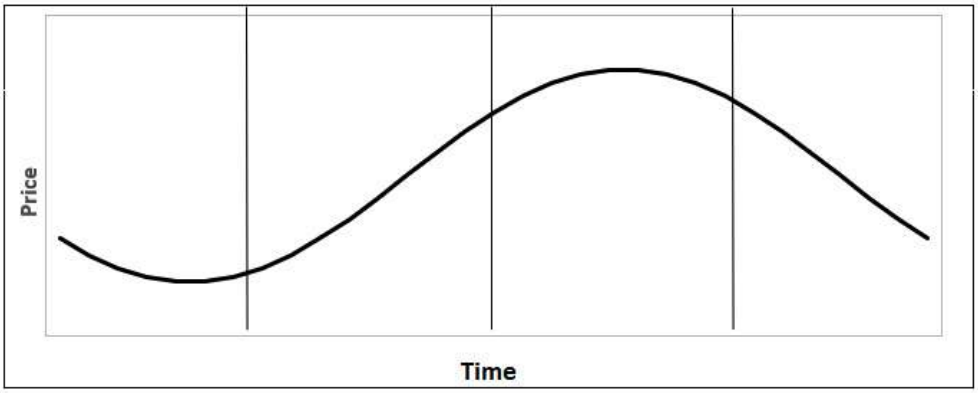
Together, prices and volumes provide the main data feed for the algorithms we are going to build.

Identifying a trend algorithmically

Many traders like to categorize the market cycles into four distinct phases (*Figure 3.4*), such as the following:

- **Accumulation**: Institutional investors quietly build positions, laying the foundation for the next uptrend
- **Uptrend**: Market sentiment is bullish as prices consistently rise, driven by increasing demand and positive investor sentiment
- **Distribution**: Institutional investors start to unwind positions, signaling the potential end of the uptrend as supply begins to surpass demand
- **Downtrend**: Market sentiment turns bearish as prices steadily decline, driven by increasing supply and waning investor confidence

We can see the four phases illustrated in the following figure:



	Accumulation	Uptrend	Distribution	Downtrend
Information	There is some good news not yet spread out	The good news spread out	There is some bad news not yet spread out	The bad news spread out
Institutional Investors	Slowly and silently buy assets	Buy/reduce	Slowly and silently liquidate their assets	Out of the market
Retail Investors	Out of the market	Buy	Buy	Sell/reduce
Prices / Volumes	Price moves sideways with low volatility, volumes are low	Price moves UP, volatility and volumes increase	Price moves sideways with low volatility, volumes are low	Price moves sharply down, volatility and volumes increase

Figure 3.4 - Market phases

As you can see in *Figure 3.4*, during the uptrend and downtrend phases, the price moves directionally and the volatility increases. On the contrary, in the accumulation and distribution phases, the price moves sideways and the volatility is low.

As algorithmic traders, our purpose is to catch such market moves using algorithms that are customized based on specific market phases. In this chapter, we are going to focus on the uptrend phase. The accumulation and distribution phases will be the subjects of *Chapter 5*. As a rule of thumb, the same concept from uptrends applies to downtrends as well, only with opposite rules. The downtrend phase will be the subject of *Chapter 6*.

Among the many techniques that are available to catch an uptrend algorithmically, we will be looking at two basic approaches:

- Moving averages
- Higher highs

Both moving averages and high breaks are very popular ways to detect visible price movements and can be easily used algorithmically. Let's see how.

Moving averages

In *Chapter 2*, we learned how to write a **Simple Moving Average** (**SMA**) indicator and apply it to a chart.

There are many moving average types available, and day by day, new ones are coming out with slightly different formulas. However, as you will see, all of them behave roughly the same. From my experience, different moving average types can help visual traders and technical analysts customize their charts and gain confidence in opening trades by looking at something they are familiar with. However, as you will see yourself, there is almost no difference between the many moving averages available when detecting a trend algorithmically.

Let's focus on moving averages' lengths for a moment because this is a crucial aspect. As we learned in the previous section, a trend is generated by information. Therefore, we need to sync our length to the most relevant information cycles, which, in the equities markets, are related to the financial reporting process: monthly, quarterly, and yearly. So, this is the reason we are going to deal mainly with lengths of 20, 50, and 200 bars on a daily time frame.

For the purpose of this chapter, we have chosen SMA, but the same concepts will apply to any moving average type. There are three main ways to use the moving average to detect a trend:

- Close above SMA
- Fast SMA above slow SMA
- Rising SMA

In the following example, we are going to use a basic 200 moving average. Setting a 200-day moving average length means that we aim to capture price movements influenced by yearly financial report updates.

Close above SMA

An algorithm to open long positions above the SMA 200 can be written as follows:

```
Input: Len(200),TradeSize(10000);
Var: Qty(0);
Qty = TradeSize/Close;
If Close > average(Close,200) then Buy ("New Long") Qty Shares next bar open;
Input:BarsIN(20);
If BarsSinceentry = BarsIN then Sell ("BarsIN") next bar open;
```

The **`Buy ("New Long") Qty Shares next bar open`** instruction is quite self-explanatory like many instructions in EasyLanguage are. Anyway, we are asking to maintain a long position of 10,000 USD

of SPY if the close is above its 200-day moving average.

`If BarsSinceentry = BarsIN` is the instruction to sell the position. Its purpose is to check any `BarsIN` days if the entry login is still in place.

SPY is an **Exchange-Traded Fund** (**ETF**) that tracks the performance of the **Standard & Poor** (**S&P**) 500 Index, which comprises 500 of the largest publicly traded companies in the United States. In *Figure 3.5*, you can see the entry-exit signal generated by the close above SMA 200 strategy:

Figure 3.5 - Close above SMA 200; entry-exit signals

Looking at *Figure 3.5*, note that there is no trade below the red 200 moving average line. In sub-graph 2, you can see the equity line of the strategy. You can apply the built-in `Strategy_Equity` indicator or create a new indicator yourself with the following two lines:

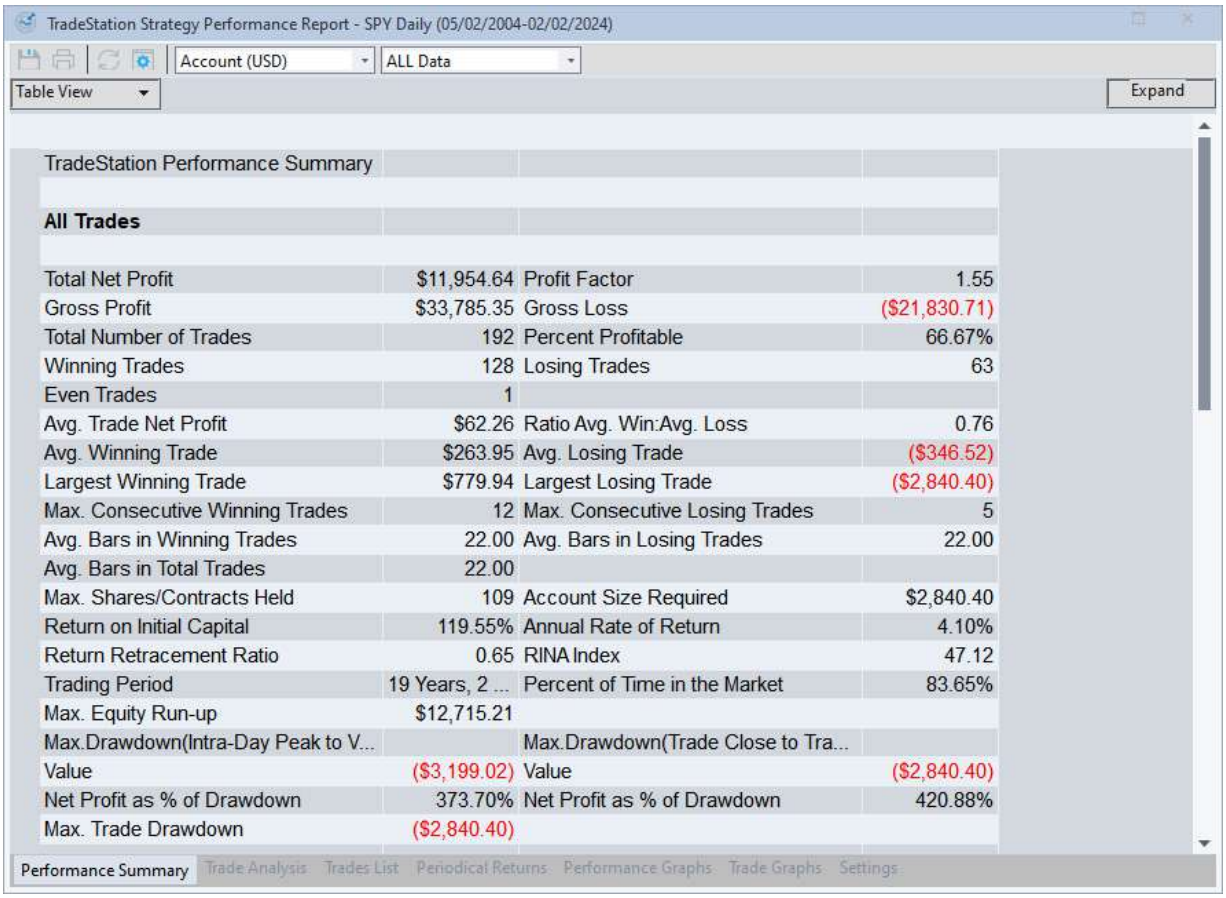
```
Plot1( I_OpenEquity, !( "OpenEquity") ,black ) ;
Plot2( I_ClosedEquity, !( "Equity") ,darkgreen ) ;
```

`I_OpenEquity` is a reserved word to refer to the open equity line, meaning the cumulative profit and loss of the strategy *including* open positions. `I_ClosedEquity` is a reserved word to refer to the closed equity line, which is the cumulative profit and loss *excluding* the still-open trade.

NOTE

Both `I_OpenEquity` and `I_ClosedEquity` only work if there is a strategy applied to the chart.

In *Figure 3.6*, you can see the TradeStation Performance Summary report:



TradeStation Performance Summary

All Trades

Total Net Profit	$11,954.64	Profit Factor	1.55
Gross Profit	$33,785.35	Gross Loss	($21,830.71)
Total Number of Trades	192	Percent Profitable	66.67%
Winning Trades	128	Losing Trades	63
Even Trades	1		
Avg. Trade Net Profit	$62.26	Ratio Avg. Win:Avg. Loss	0.76
Avg. Winning Trade	$263.95	Avg. Losing Trade	($346.52)
Largest Winning Trade	$779.94	Largest Losing Trade	($2,840.40)
Max. Consecutive Winning Trades	12	Max. Consecutive Losing Trades	5
Avg. Bars in Winning Trades	22.00	Avg. Bars in Losing Trades	22.00
Avg. Bars in Total Trades	22.00		
Max. Shares/Contracts Held	109	Account Size Required	$2,840.40
Return on Initial Capital	119.55%	Annual Rate of Return	4.10%
Return Retracement Ratio	0.65	RINA Index	47.12
Trading Period	19 Years, 2 ...	Percent of Time in the Market	83.65%
Max. Equity Run-up	$12,715.21		
Max.Drawdown(Intra-Day Peak to V...		Max.Drawdown(Trade Close to Tra...	
Value	($3,199.02)	Value	($2,840.40)
Net Profit as % of Drawdown	373.70%	Net Profit as % of Drawdown	420.88%
Max. Trade Drawdown	($2,840.40)		

Figure 3.6 - The TradeStation Performance Summary

Performance Summary is a report that is automatically generated by TradeStation to provide an overview of the strategy's performance. In *Table 3.1*, we can see the main key performance indicators that are calculated by TradeStation:

Key performance indicator	**Description**	**Abbreviation**
Total net profit	Net profit or loss during the specified period	Profit
Gross profit	Cumulative total of all winning trades	GrossProfit
Gross loss	Cumulative total of all losing trades	GrossLoss
Total number of trades	Total number of completed trades	TotTrades

Key performance indicator	**Description**	**Abbreviation**
Winning trades	Total number of completed profitable trades	WinTrades
Losing trades	Total number of completed unprofitable trades	LosTrades
Percent profitable	Percentage of completed trades that were profitable	HitRate% = WinTrades / TotTrades
Average trade net profit	Average amount of money (won or loss) for all completed trades	AvgT = Profit / TotTrades
Average winning trade	Average amount of money (won) for all completed profitable trades	AvgW = GrossProfit / WinTrades
Average losing trade	Average amount of money (lost) for all completed unprofitable trades	AvgL = GrossLoss / LosTrades
Maximum drawdown (intra-day peak to valley)	Greatest loss drawdown, from the previous highest-equity run-up, bar to bar looking across all trades	MaxDD
Ratio average win:average loss	Ratio on average of how much you won compared to the amount you lost	AvgRato = Avg W / AvgL
Net profit as % of drawdown	Reward/risk ratio between the total profit and the max drawdown peak to valley	RewRisk = NetProfit / MaxDD *100

Table 3.1 - Key performance indicators

As you can see in *Table 3.1*, there are many (maybe too many) key performance indicators, as calculated in the TradeStation Performance Summary report. Among the many key performance

indicators available, we will focus on four of them in this book in order to make the evaluation process straightforward:

- Number of trades
- Total net profit ($)
- Max drawdown ($)
- Reward/risk ratio (net profit as a percentage of max drawdown)

So, for example, the *Figure 3.6* Performance Summary report can be synthesized as follows:

- 192 trades
- 11.954,64 USD net profit
- -3.199.02 USD max drawdown
- 373.70% reward/risk ratio (11,954,64 USD divided by -3,199.02 USD)

Let’s now move on to a different trend technique consisting of two moving averages.

Fast SMA above slow SMA

This is a strategy to open a 10,000 USD long position if the 20-day SMA is above the 200-day SMA, and hold the position for 20 days:

```
Input: FastLen(20),SlowLen(200),TradeSize(10000);
Var: Qty(0);
Qty = TradeSize/Close;
If average(Close,FastLen) > average(Close,SlowLen) then Buy ("New Long") Qty Shares next
bar open;
Input:BarsIN(20);
If BarsSinceentry = BarsIN then Sell ("BarsIN") next bar open;
```

As you can see in the preceding code, we only changed the conditional instruction:

`average(Close,FastLen) > average(Close,SlowLen).`

In *Figure 3.7*, you can see the entry-exit signal generated by the strategy:

Figure 3.7 - The fast SMA above slow SMA strategy — entry-exit signals

As you can see in *Figure 3.7*, entry signals are now generated only when the thin line (the fast SMA) is above the tick line (the slow SMA).

In *Figure 3.8*, you can see the related performance summary.

TradeStation Strategy Performance Report - SPY Daily (05/02/2004-02/02/2024)

Account (USD) | ALL Data | Table View | Expand

TradeStation Performance Summary			
All Trades			
Total Net Profit	$13,254.66	Profit Factor	1.67
Gross Profit	$32,960.50	Gross Loss	($19,705.84)
Total Number of Trades	183	Percent Profitable	65.03%
Winning Trades	119	Losing Trades	64
Even Trades	0		
Avg. Trade Net Profit	$72.43	Ratio Avg. Win:Avg. Loss	0.90
Avg. Winning Trade	$276.98	Avg. Losing Trade	($307.90)
Largest Winning Trade	$1,394.22	Largest Losing Trade	($1,280.10)
Max. Consecutive Winning Trades	14	Max. Consecutive Losing Trades	6
Avg. Bars in Winning Trades	22.00	Avg. Bars in Losing Trades	22.00
Avg. Bars in Total Trades	22.00		
Max. Shares/Contracts Held	114	Account Size Required	$2,311.16
Return on Initial Capital	13.25%	Annual Rate of Return	0.65%
Return Retracement Ratio	0.84	RINA Index	53.26
Trading Period	19 Years, 2 ...	Percent of Time in the Market	79.46%
Max. Equity Run-up	$13,744.48		
Max.Drawdown(Intra-Day Peak to V...		Max.Drawdown(Trade Close to Tra...	
Value	($3,985.27)	Value	($2,311.16)
Net Profit as % of Drawdown	332.59%	Net Profit as % of Drawdown	573.51%
Max. Trade Drawdown	($2,718.30)		

Performance Summary | Trade Analysis | Trades List | Periodical Returns | Performance Graphs | Trade Graphs | Settings

Figure 3.8 - The fast SMA above slow SMA strategy — Performance Summary

The performance summary in *Figure 3.8* can be recapped as follows:

- 183 trades
- $ 13,254,66 net profit
- $ -3,985.27 max drawdown
- 332.59% reward/risk ratio

Let’s move on now to a different way of using the moving average to identify a trend: the rising SMA.

Rising SMA

This is a strategy to open a 10,000 USD long position if the 200-day moving average is rising and hold the position for 20 days:

```
Input: Len(20),TradeSize(10000);
Var: Qty(0);
Qty = TradeSize/Close;
If average(Close,Len) > average(Close,Len)[1] then Buy ("New Long") Qty Shares next bar
open;
Input:BarsIN(20);
If BarsSinceentry = BarsIN then Sell ("BarsIN") next bar open;
```

Comparing this to the previous code block, we only changed the conditional instruction: **`average(Close,Len) > average(Close,Len)[1]`**.

In *Figure 3.9*, you can see the entry-exit signals generated by the rising SMA strategy:

Figure 3.9 - The rising SMA strategy — entry-exit signals

Looking at *Figure 3.9*, in order to check whether our script is correct, we can check whether entry signals correspond with a rising moving average line. In *Figure 3.10*, you can see the rising SMA strategy performance summary:

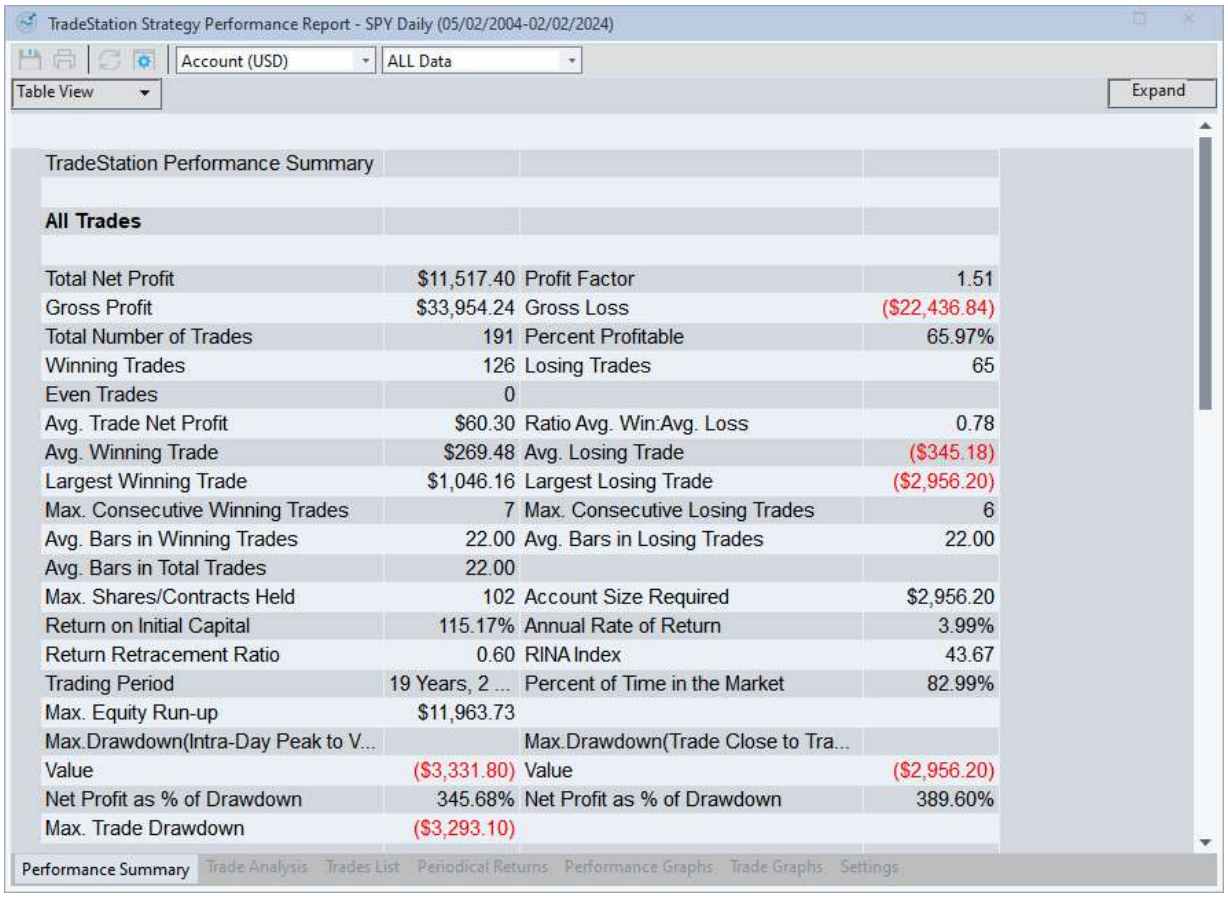

TradeStation Strategy Performance Report - SPY Daily (05/02/2004-02/02/2024)
Account (USD)
ALL Data
Table View
Expand
TradeStation Performance Summary
All Trades
Total Net Profit $11,517.40 Profit Factor 1.51
Gross Profit $33,954.24 Gross Loss ($22,436.84)
Total Number of Trades 191 Percent Profitable 65.97%
Winning Trades 126 Losing Trades 65
Even Trades 0
Avg. Trade Net Profit $60.30 Ratio Avg. Win:Avg. Loss 0.78
Avg. Winning Trade $269.48 Avg. Losing Trade ($345.18)
Largest Winning Trade $1,046.16 Largest Losing Trade ($2,956.20)
Max. Consecutive Winning Trades 7 Max. Consecutive Losing Trades 6
Avg. Bars in Winning Trades 22.00 Avg. Bars in Losing Trades 22.00
Avg. Bars in Total Trades 22.00
Max. Shares/Contracts Held 102 Account Size Required $2,956.20
Return on Initial Capital 115.17% Annual Rate of Return 3.99%
Return Retracement Ratio 0.60 RINA Index 43.67
Trading Period 19 Years, 2 ... Percent of Time in the Market 82.99%
Max. Equity Run-up $11,963.73
Max.Drawdown(Intra-Day Peak to V... Max.Drawdown(Trade Close to Tra...
Value ($3,331.80) Value ($2,956.20)
Net Profit as % of Drawdown 345.68% Net Profit as % of Drawdown 389.60%
Max. Trade Drawdown ($3,293.10)
Performance Summary
Trade Analysis
Trades List
Periodical Returns
Performance Graphs
Trade Graphs
Settings

Figure 3.10 - The rising SMA strategy — Performance Summary

The performance summary in *Figure 3.10* can be summarized as follows:

- 191 trades
- $ 11,517.40 net profit
- $ -3,331.80 max drawdown
- 345.68% reward/risk ratio

Let’s move on to a different approach than moving averages to catch uptrends: higher highs.

Higher highs

When there is a noticeable positive trend, prices make new highs. In order to catch algorithmically new highs, there are two options:

- A 200-day high higher than 20 days before
- A 50-day high higher than 10 days before

A 200-day high higher than 20 days before

Let’s first write our indicator to visualize such trading ideas. In order to visualize higher highs, we can write the following:

```
Input:BarsBack(200),Delay(20);
var: PreviousTop (0),NewTop(0);
NewTop      = highest(High , BarsBack);
PreviousTop = highest(High , BarsBack)[delay];
plot1(NewTop,"NewTop");
plot2(PreviousTop,"PreviousTop");
```

This indicator will plot two lines representing **`NewTop`** and **`PreviousTop`**:

- **`NewTop`** is the highest value of the highs for the last **`BarsBack`** bars
- PreviousTop is determined by referring back to the **`NewTop`** from **`[delay]`** bars ago

> ***NOTE***
>
> *As you will remember from Chapter 2, square brackets are used to reference past data.*

Let’s move on to the strategy now. Assuming that we want to open a 10,000 USD long position if there is a new high in the last 20 bars, we can write the following:

```
Input:BarsBack(200),Delay(20);
var: PreviousTop (0),NewTop(0);
NewTop      = highest(High , BarsBack);
PreviousTop = highest(High , BarsBack)[Delay];
Input:TradeSize(10000);
Var: Qty(0);
Qty = TradeSize/Close;
If NewTop >= PreviousTop then Buy ("New Long") Qty Shares next bar open;
If NewTop <  PreviousTop then Sell("Sell")
next bar open;
```

> ***NOTE***
>
> *In order to reduce typos, you can start writing this strategy by copying and pasting the first four lines of the previous indicator and then adding the order instructions.*

In *Figure 3.11*, you can see a chart analysis with both the indicator and the strategy applied to it:

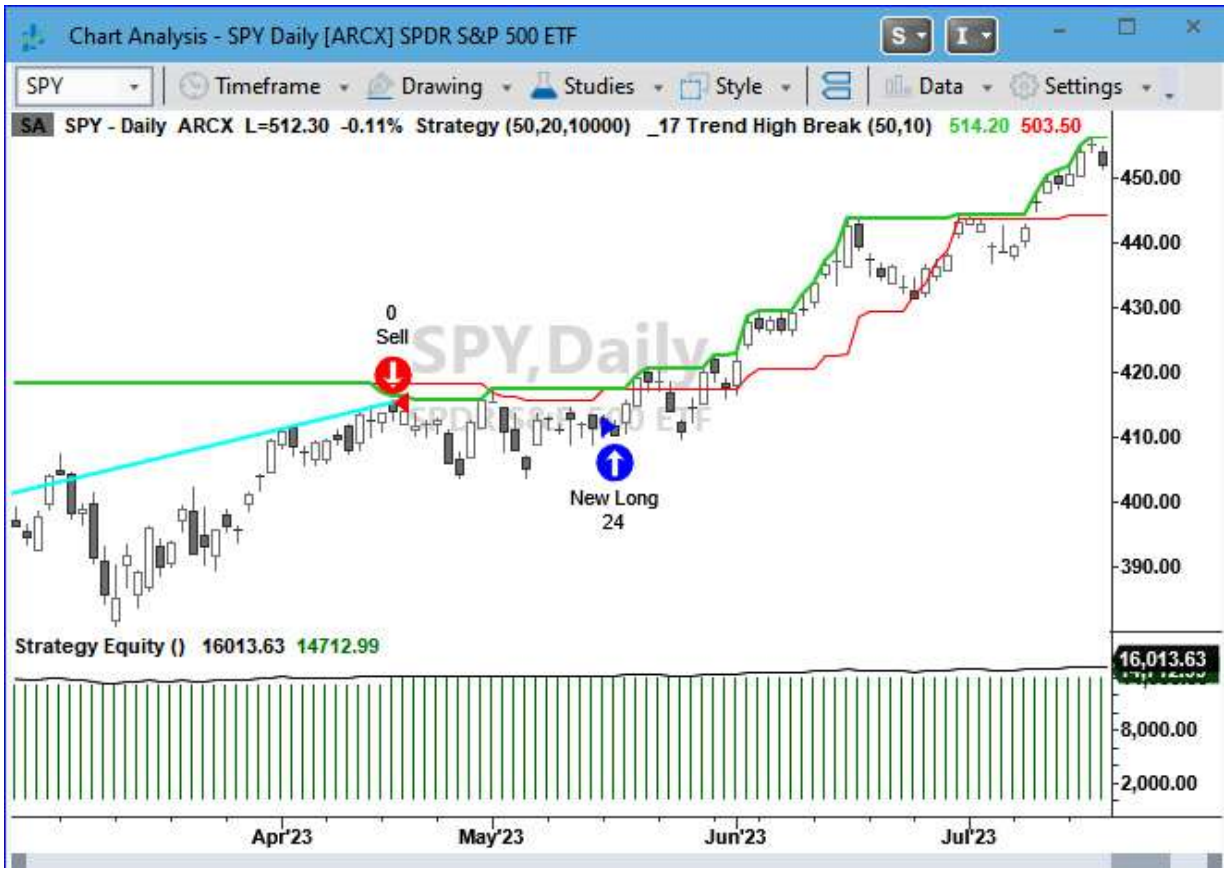
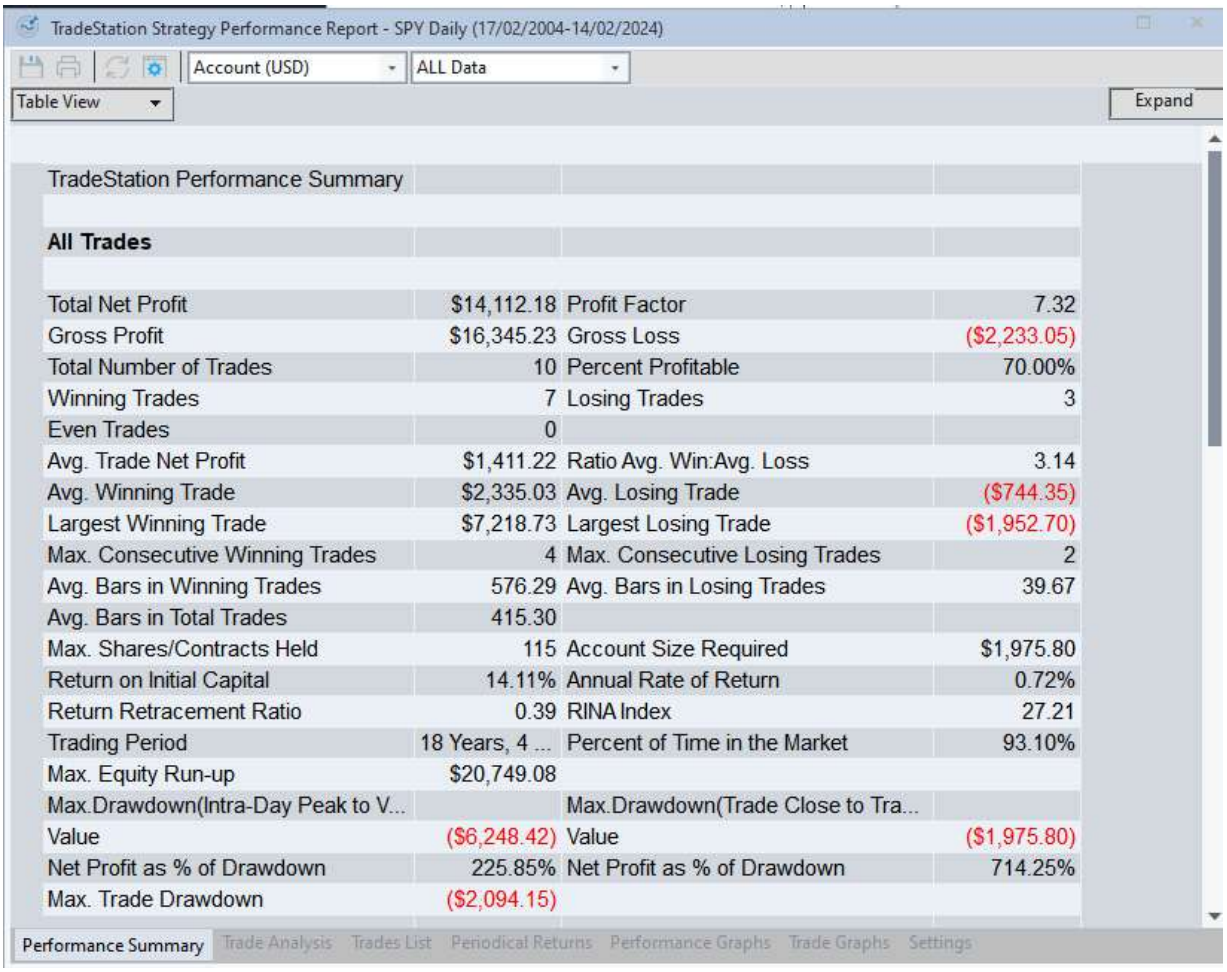


Figure 3.11 - The higher highs strategy — entry-exit signals

In *Figure 3.11*, we can see that a long position is opened any time the thin red line representing **`NewTop`** is equal or greater to the tick green line representing **`PreviousTop`**. In *Figure 3.12*, you can see the related performance summary:



TradeStation Performance Summary

All Trades

Total Net Profit	$14,112.18	Profit Factor	7.32
Gross Profit	$16,345.23	Gross Loss	($2,233.05)
Total Number of Trades	10	Percent Profitable	70.00%
Winning Trades	7	Losing Trades	3
Even Trades	0		
Avg. Trade Net Profit	$1,411.22	Ratio Avg. Win:Avg. Loss	3.14
Avg. Winning Trade	$2,335.03	Avg. Losing Trade	($744.35)
Largest Winning Trade	$7,218.73	Largest Losing Trade	($1,952.70)
Max. Consecutive Winning Trades	4	Max. Consecutive Losing Trades	2
Avg. Bars in Winning Trades	576.29	Avg. Bars in Losing Trades	39.67
Avg. Bars in Total Trades	415.30		
Max. Shares/Contracts Held	115	Account Size Required	$1,975.80
Return on Initial Capital	14.11%	Annual Rate of Return	0.72%
Return Retracement Ratio	0.39	RINA Index	27.21
Trading Period	18 Years, 4 ...	Percent of Time in the Market	93.10%
Max. Equity Run-up	$20,749.08		
Max.Drawdown(Intra-Day Peak to V...		Max.Drawdown(Trade Close to Tra...	
Value	($6,248.42)	Value	($1,975.80)
Net Profit as % of Drawdown	225.85%	Net Profit as % of Drawdown	714.25%
Max. Trade Drawdown	($2,094.15)		

Figure 3.12 - The higher highs (20-200) strategy — Performance Summary

The performance summary in *Figure 3.12* can be summarized as follows:

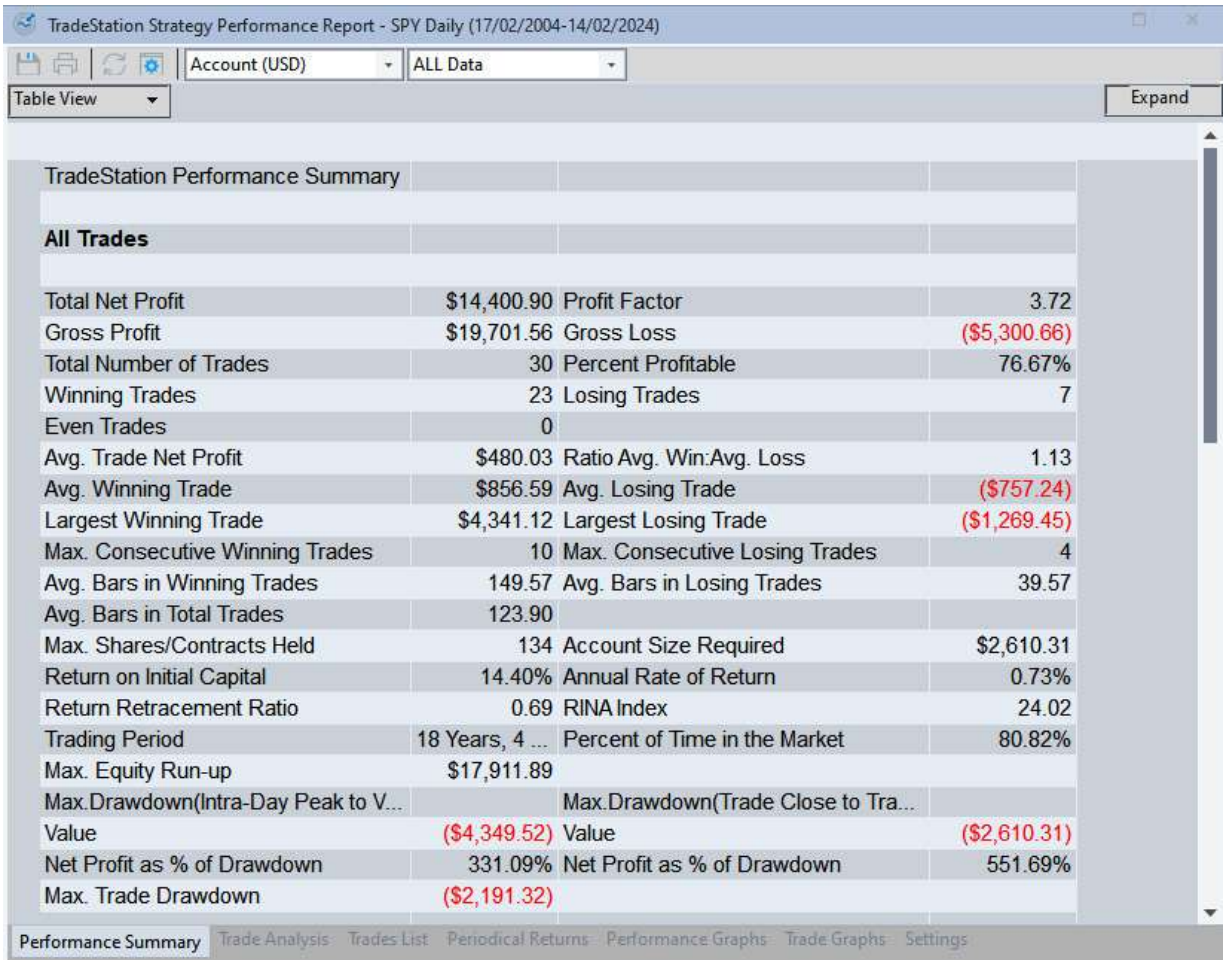
- 10 trades
- $ 14,112,18 net profit
- $ -6,248.42 max drawdown
- 225.95% reward/risk ratio

A 50-day high higher than 10 days before

Let’s now reduce the settings to see what will happen if we move from a yearly based trend logic to a quarterly based one. From the chart’s menu, go to **Studies** | **Edit Strategies** | **Customize** | **Inputs** and set the following settings:

- Set **BarsBack** to **`50`**
- Set **Delay** to **`10`**

The strategy will automatically be re-calculated and produce the performance report shown in *Figure 3.13*:



TradeStation Performance Summary			
All Trades			
Total Net Profit	$14,400.90	Profit Factor	3.72
Gross Profit	$19,701.56	Gross Loss	($5,300.66)
Total Number of Trades	30	Percent Profitable	76.67%
Winning Trades	23	Losing Trades	7
Even Trades	0		
Avg. Trade Net Profit	$480.03	Ratio Avg. Win:Avg. Loss	1.13
Avg. Winning Trade	$856.59	Avg. Losing Trade	($757.24)
Largest Winning Trade	$4,341.12	Largest Losing Trade	($1,269.45)
Max. Consecutive Winning Trades	10	Max. Consecutive Losing Trades	4
Avg. Bars in Winning Trades	149.57	Avg. Bars in Losing Trades	39.57
Avg. Bars in Total Trades	123.90		
Max. Shares/Contracts Held	134	Account Size Required	$2,610.31
Return on Initial Capital	14.40%	Annual Rate of Return	0.73%
Return Retracement Ratio	0.69	RINA Index	24.02
Trading Period	18 Years, 4 ...	Percent of Time in the Market	80.82%
Max. Equity Run-up	$17,911.89		
Max.Drawdown(Intra-Day Peak to V...		Max.Drawdown(Trade Close to Tra...	
Value	($4,349.52)	Value	($2,610.31)
Net Profit as % of Drawdown	331.09%	Net Profit as % of Drawdown	551.69%
Max. Trade Drawdown	($2,191.32)		

Figure 3.13 - The higher highs (50-20) performance summary

Figure 3.13 can be summarized as follows:

- 30 trades
- $ 14,400,90 net profit
- $ -4,349.52 max drawdown
- 331.09% reward/risk ratio

Let’s now compare the five trend techniques we have developed so far by collating the four main key performance indicators we have tracked into one table as in *Table 3.1*:

Strategy	**Net profit (USD)**	**Total trades**	**Max drawdown (USD)**	**Reward/risk ratio**

Strategy	Net profit (USD)	Total trades	Max drawdown (USD)	Reward/risk ratio
Close above SMA 200	11,954.64	192	-3,199.02	**373.70%**
SMA 20 above SMA 200	13,254.66	183	-3,095.27	332.59%
SMA 200 rising	11,517.40	191	**-3,331.80**	345.68%
Higher highs (200,20)	14,112.18	10	-6,243.42	225.85%
Higher highs (50,10)	**14,400.90**	30	-4,349.52	331.09%

Table 3.2 - Strategies comparison

By analyzing *Table 3.2*, we can make the following evaluations:

- The higher highs (50,10) technique shows the highest net profit ($14,400.90)
- The SMA rising technique shows the lowest max drawdown ($-3,331.80)
- The close above SMA 200 technique shows the highest reward/risk ratio (373.70%)

All traders will have their preferred key performance indicators to evaluate strategy. As you are going to analyze hundreds of different strategies, the key is to choose just one of them. For the purposes of this book, we have chosen the reward/risk ratio. So, we can conclude that, so far, *the best strategy for the example in this chapter is the close above 200 SMA with a reward/risk ratio of 373.70%*.

However, entering and exiting around a moving average due to market volatility might make us trade too many times. So, the question is: can we filter trades without altering performances? In other words: can we avoid **market noise**?

Handling market noise

Market noise refers to the random fluctuations and irrelevant information that can obscure the underlying trends or signals in financial markets. It can include various factors, such as rumors, news headlines, speculative trading, and other short-term market movements that are not based on fundamental factors or long-term trends.

There are two different ways to address the noise issue:

- Entry confirmations
- Volatility bands

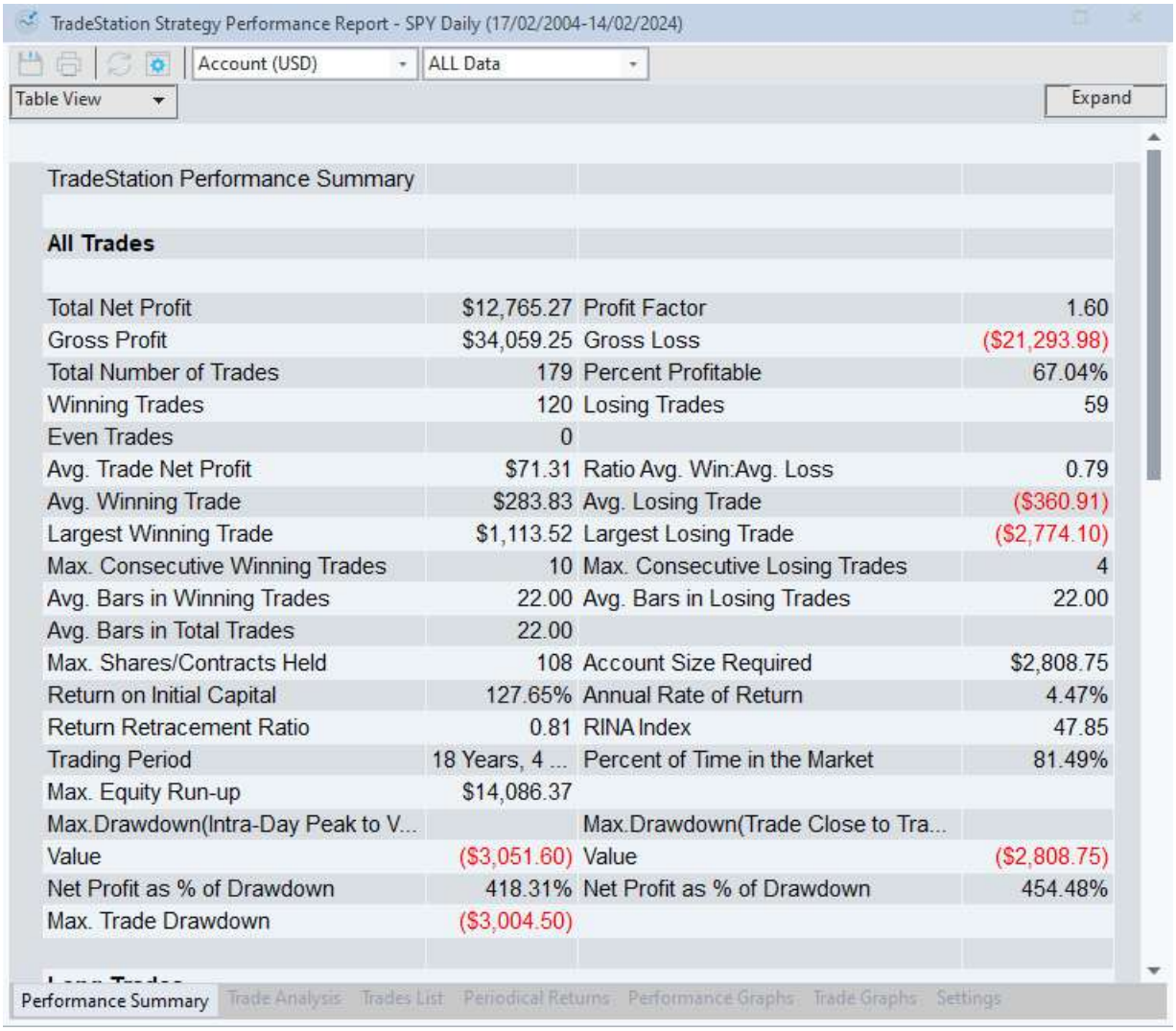
Entry confirmations

We can ask the system to wait for a confirmation before opening a new trade. A confirmation could be a close above the moving average for two consecutive days, so we can write the following:

```
EntryCondition=Close>average(close,Len) and Close[1]>average(Close,Len)[1];
```

In this example, our entry condition will be both the current close above the current moving average value and the previous close above the previous moving average.

In *Figure 3.14*, we can see the performance report with entry confirmation:



TradeStation Performance Summary

All Trades

Total Net Profit	$12,765.27	Profit Factor	1.60
Gross Profit	$34,059.25	Gross Loss	($21,293.98)
Total Number of Trades	179	Percent Profitable	67.04%
Winning Trades	120	Losing Trades	59
Even Trades	0		
Avg. Trade Net Profit	$71.31	Ratio Avg. Win:Avg. Loss	0.79
Avg. Winning Trade	$283.83	Avg. Losing Trade	($360.91)
Largest Winning Trade	$1,113.52	Largest Losing Trade	($2,774.10)
Max. Consecutive Winning Trades	10	Max. Consecutive Losing Trades	4
Avg. Bars in Winning Trades	22.00	Avg. Bars in Losing Trades	22.00
Avg. Bars in Total Trades	22.00		
Max. Shares/Contracts Held	108	Account Size Required	$2,808.75
Return on Initial Capital	127.65%	Annual Rate of Return	4.47%
Return Retracement Ratio	0.81	RINA Index	47.85
Trading Period	18 Years, 4 ...	Percent of Time in the Market	81.49%
Max. Equity Run-up	$14,086.37		
Max.Drawdown(Intra-Day Peak to V...		Max.Drawdown(Trade Close to Tra...	
Value	($3,051.60)	Value	($2,808.75)
Net Profit as % of Drawdown	418.31%	Net Profit as % of Drawdown	454.48%
Max. Trade Drawdown	($3,004.50)		

Figure 3.14 - Entry confirmation to filter market noise — Performance Summary

The Performance Summary report in *Figure 3.14* can be recapped as follows:

- 179 trades
- $ 12,765.27 net profit

- $ -3,051.80 max drawdown
- 418.31% reward/risk ratio

Please note how the reward/risk ratio increased from 373.70% (*Table 3.2* to 418.31%, meaning that filtering noise makes sense to improve the strategy results.

Volatility bands

Another way to filter market noise is by adding the moving average volatility band. A volatility band can be obtained by adding (or subtracting) a certain number of standard deviations to the moving average, where the standard deviation is a statistical measure of the dispersion or variability of a set of data points around the mean (average) of the dataset. It indicates how much individual data points differ from the mean.

The formula to calculate the standard deviation is shown in *Figure 3.15*:

$$StdDev = \sqrt{\frac{\Sigma(X-\mu)^2}{N}}$$

X = The Value in the data distribution
μ = The Population Mean
N = Total Number of Observations

Figure 3.15 - The standard deviation formula

However, don’t worry if you are not familiar with this statistical calculation because TradeStation provides a built-in function for standard deviation.

So, in order to filter noise with volatility bands, we can write the following:

```
if NoiseFilterType=2 then EntryCondition =
 Close >average(close,200) + 0.5*StdDev(close,200);
```

`StdDev` is the built-in function. The performance report back-testing the volatility filter is shown in *Figure 3.16*:

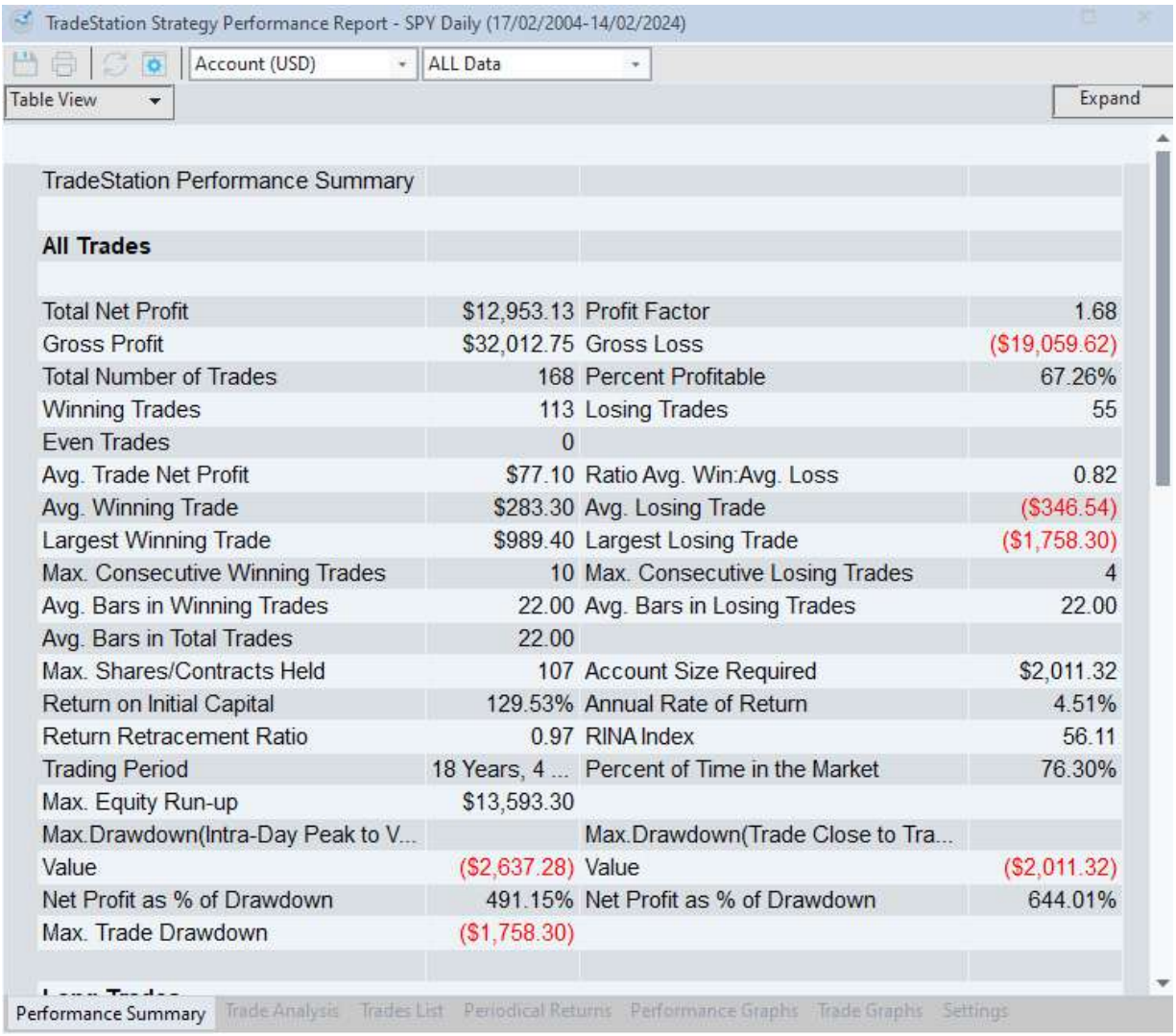

TradeStation Strategy Performance Report - SPY Daily (17/02/2004-14/02/2024)
Account (USD)
ALL Data
Table View
Expand
TradeStation Performance Summary
All Trades
Total Net Profit $12,953.13 Profit Factor 1.68
Gross Profit $32,012.75 Gross Loss ($19,059.62)
Total Number of Trades 168 Percent Profitable 67.26%
Winning Trades 113 Losing Trades 55
Even Trades 0
Avg. Trade Net Profit $77.10 Ratio Avg. Win:Avg. Loss 0.82
Avg. Winning Trade $283.30 Avg. Losing Trade ($346.54)
Largest Winning Trade $989.40 Largest Losing Trade ($1,758.30)
Max. Consecutive Winning Trades 10 Max. Consecutive Losing Trades 4
Avg. Bars in Winning Trades 22.00 Avg. Bars in Losing Trades 22.00
Avg. Bars in Total Trades 22.00
Max. Shares/Contracts Held 107 Account Size Required $2,011.32
Return on Initial Capital 129.53% Annual Rate of Return 4.51%
Return Retracement Ratio 0.97 RINA Index 56.11
Trading Period 18 Years, 4 ... Percent of Time in the Market 76.30%
Max. Equity Run-up $13,593.30
Max.Drawdown(Intra-Day Peak to V... Max.Drawdown(Trade Close to Tra...
Value ($2,637.28) Value ($2,011.32)
Net Profit as % of Drawdown 491.15% Net Profit as % of Drawdown 644.01%
Max. Trade Drawdown ($1,758.30)
Performance Summary
Trade Analysis
Trades List
Periodical Returns
Performance Graphs
Trade Graphs
Settings

Figure 3.16 - Volatility bands to filter noise — Performance Summary

The performance summary in *Figure 3.16* can be summarized as follows:

- 168 trades
- $ 12,953.13 net profit
- $ -2,637.28 max drawdown
- 491.15% reward/risk ratio

These results tell us that filtering noise with volatility bands further enhances the strategy result by improving the reward/risk ratio from 418.31% to 491.15%.

Here, you can see the full code to back-test noise filters:

```
Input: Len(200),TradeSize(10000);
Var: Qty(0);
Qty = TradeSize/Close;
Input:NoiseFilterType(1);
```

```
var:EntryCondition(false);
if NoiseFilterType=0 then EntryCondition = Close >average(Close,Len)[1];
if NoiseFilterType=1 then
EntryCondition=Close>average(close,Len) and Close[1]>average(Close,Len)[1];
if NoiseFilterType=2 then EntryCondition =
 Close >average(close,200) + 0.5*StdDev(close,200);
If EntryCondition then Buy ("New Long") Qty Shares next bar open;
Input:BarsIN(20);
If BarsSinceentry = BarsIN then Sell ("BarsIN") next bar open;
```

The reason why I added the **`Input:NoiseFilterType`** instruction is that this allows us to easily perform multiple back-tests in just one step. Let me illustrate:

1. From the main menu on the chart, go to **Studies** | **Edit Strategies** | **Customize**.
2. Click on **Noise Filter Type**.
3. Click on **Optimize**.
4. Set **Start** to **`0`**, **Stop** to **`2`**, and **Step** to **`1`**.
5. Click **OK**.
6. Click **Optimize**.

> ***NOTE***
>
> *With the **`if NoiseFilterType=0 then EntryCondition = Close >average(Close,Len)[1];`** instruction, we are ignoring the noise filter, which is our strategy benchmark.*

An optimization process will start. TradeStation will automatically back-test the three different filter noises and produce the strategy optimization report shown in *Figure 3.17*.

	19_Trend_close_above_SMA Noise: NoiseFilterType	All: Net Profit	All: Total Trades	All: Max Intraday Drawdown	All: Return on Account
1	0	11,328.43	182	-3,102.42	365.15
2	1	12,765.28	179	-3,004.50	424.87
3	2	12,953.13	168	-2,621.32	494.15

Figure 3.17 - The strategy optimization report with three different noise filters

In *Figure 3.17*, you can see a recap of what we have analyzed so far. If you look at the first column, you can see the different noise filters applied, as follows:

- **0** for no noise filter
- **1** for two closes above the moving average
- **2** for a close above the upper volatility band

If you look at the `NetProfit` column, you will see that `NoiseFilterType=2` is the one showing the highest net profit.

If you look at the last column on the right, you will see that, again, `NoiseFilterType=2` is the one showing the highest reward/risk ratio.

> ***NOTE***
>
> *TradeStation often refers to the reward/risk ratio with the term return on account. Anyway, we are talking about the ratio between the net profit and the max drawdown.*

Summary

In this chapter, we defined the market rationale behind a trend, the role of information, and the market phases theory. We then went through two different approaches to algorithmically catch a trend, one based on moving averages and the other on higher highs. We then back-tested five different strategies and compared results across four different key performance metrics. Then, we discussed two ways to filter market noise: we wrote relative scripts and back-tested different filters. The main purpose of this chapter was to explore the process of transforming trading ideas into algorithms. In the next chapter, we are going to delve more into back-testing and optimization processes.

4

Strategy Backtesting and Validation

In the previous chapter, we saw how the process of transforming a trading idea into a trading algorithm works. In this chapter, we are going to delve into strategy evaluation by learning about using **EasyLanguage** code to run backtesting and using TradeStation and Excel features as a unique tool for strategy evaluation purposes.

The main purpose of this chapter is for you to understand both the financial and statistical aspects of algorithmic trading.

In this chapter, we will cover the following topics:

- Understanding backtesting and overfitting
- Sensitivity analysis
- Backtesting across different symbols
- Backtesting versus buy-and-hold
- In-sample and out-of-sample

Understanding backtesting and overfitting

Backtesting is a crucial tool in the arsenal of traders, investors, and financial analysts. It involves testing a trading strategy or investment hypothesis using historical data to assess its viability and performance. Any time we evaluate an algorithmic trading strategy, we need to answer two questions:

- Does the strategy perform on historical data?
- Will the strategy perform on future data?

The tools to answer the first question are straightforward. We can launch TradeStation backtesting and evaluate results by analyzing the financial metrics we described in *Chapter 3*.

Answering the second question is a bit more complicated as there is no built-in tool to achieve such a task and the research for this crucial aspect of algorithmic trading is still ongoing.

As you'll remember from the previous chapter (*Figure 3.1*), we said that the development process of a trading strategy is iterative, where you refine and improve the strategy based on the insights gained from each round of testing. This may involve adjusting the strategy's rules many times. The iterative process is crucial for identifying key parameters of algorithmic strategies, but it presents a downside: overfitting.

Overfitting is the main enemy of algorithmic traders; it occurs when a model is overly complex and fits too closely to the historical data, including random noise or fluctuations present in the data. In other words, the model fits the past data so closely that it loses the ability to generalize and make successful trades on new, unseen data.

To better understand overfitting and how to avoid it, let's look into the following concepts:

- Strategy complexity
- Strategy robustness

Strategy complexity

Strategy complexity refers to the intricacy and sophistication of the strategic plan or approach adopted by the trader; a complex strategy might involve numerous variables, interdependencies, and considerations that require careful analysis and implementation.

Strategy robustness

Strategy robustness, on the other hand, relates to the resilience and adaptability of a strategy in the face of uncertainties, disruptions, and unexpected events; a robust strategy is designed to withstand shocks, changes, and unforeseen circumstances without significant loss of effectiveness or coherence. Robust strategies typically incorporate flexibility, redundancy, and contingency planning to mitigate risks and capitalize on opportunities. The focus of robustness is on creating strategies that are durable and can perform well across a range of potential scenarios, rather than optimizing solely for a specific set of conditions.

In *Figure 4.1*, you can see the relationship between strategy complexity and strategy robustness.

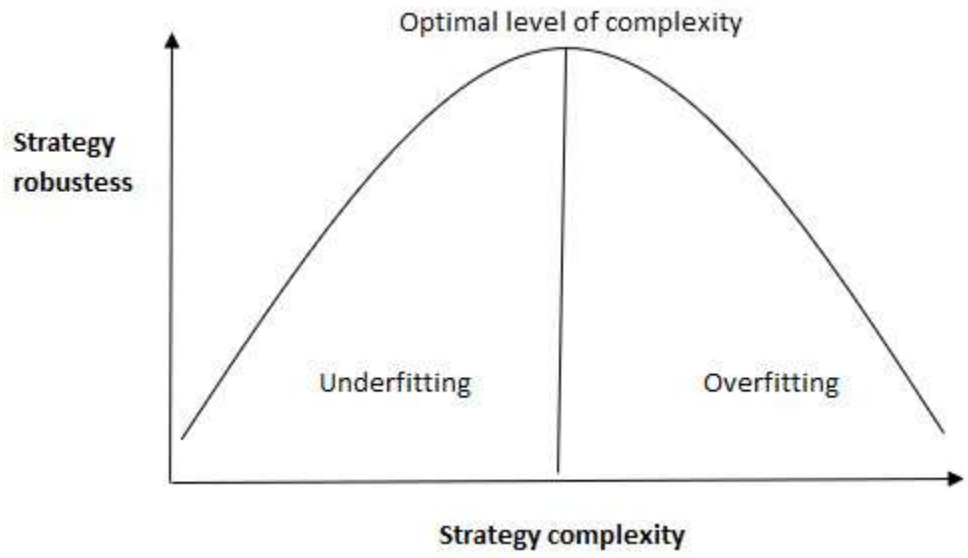


Figure 4.1 – Strategy complexity versus strategy robustness

In *Figure 4.1*, you can see how **strategy robustness** can increase with **strategy complexity** up to a certain threshold that we can define as the **optimal level of complexity**; further increases in complexity beyond such a level might lead to diminishing robustness. As a strategy becomes more complex, it tends to encompass a broader range of potential scenarios, risks, and contingencies; anyway, beyond the optimal level of complexity, the strategy may become overly tailored to the historical data it's based on, resulting in diminished performance when applied to new, unseen data.

In my experience, the best way to avoid overfitting is to keep any trading strategy as simple as possible. This involves drastically reducing the number of features in the strategy, selecting parameters based solely on solid rationales, and, as discussed in *Chapter 3*, having a clear reason for their usage, which should be explainable in plain English.

Let's now move on to how to practically run backtesting with TradeStation and reduce the risk of overfitting.

Sensitivity analysis

Sensitivity analysis is used in various fields, including finance, engineering, economics, and statistics, to understand how changes in certain variables affect the outcome of a model or decision-making process. It helps in assessing the robustness and reliability of a model by identifying the most critical factors influencing its results.

We are now going to present three sensitivity analysis techniques:

- **One-way sensitivity analysis**: This involves varying one input variable while keeping all other variables constant. It helps to understand the impact of changes in a single parameter on the model's output.

- **Double-way sensitivity analysis**: In this approach, two input variables are varied simultaneously to assess their combined impact on the output.
- **Multiple-way sensitivity analysis**: You can use this technique if you want to analyze more than two input variables simultaneously.

One-way sensitivity analysis

Let's go back to the trend strategy we wrote in *Chapter 3*. The strategy has two main parameters: `FastLen` and `SlowLen`. Let's modify such parameters one at a time starting with `FastLen`.

From the menu on top of the chart, go to **Studies** | **Edit Strategies** | **Customize** | **Inputs** and set the `FastLen` settings as shown in *Figure 4.2*:

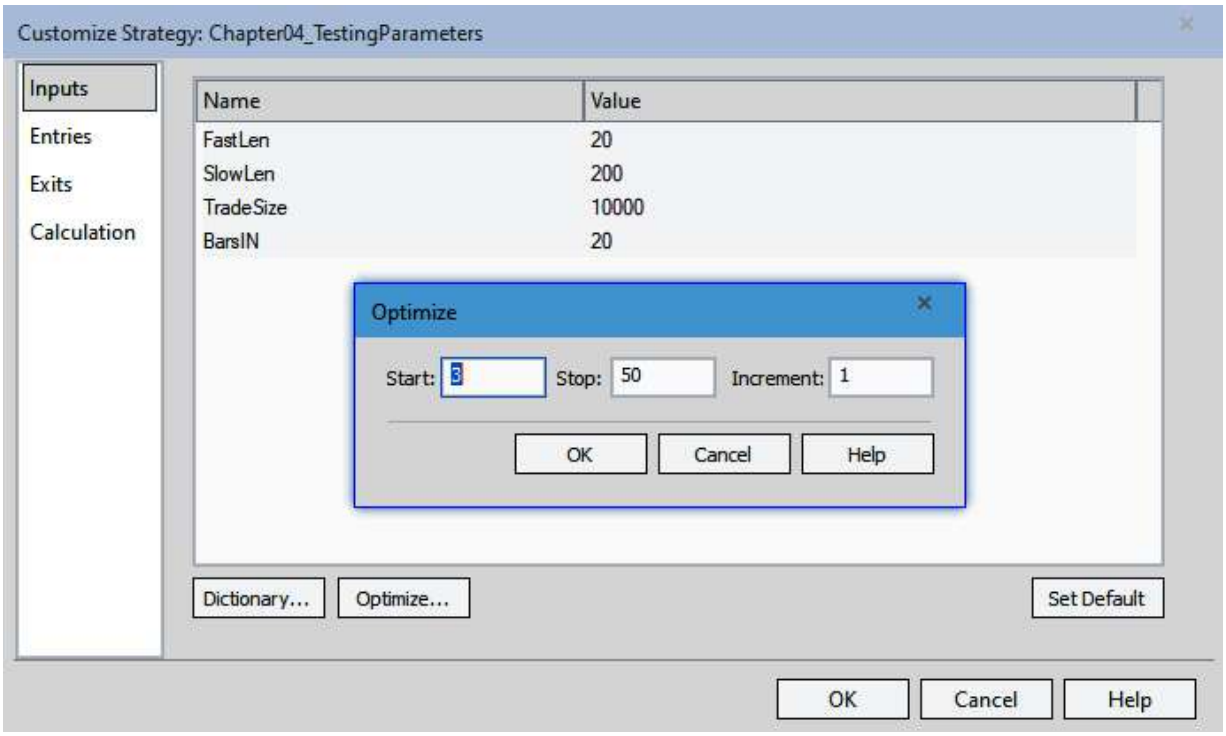


Figure 4.2 – Optimizing FastLen

In *Figure 4.2*, you can see how to vary the `FastLen` input from `3` to `50` with an increment of `1`; by running this optimization, TradeStation will produce a report like *Figure 4.3*:

TestingF FastLen	All: Total Trades	All: Net Profit	All: Max Intraday Drawdown	All: Return on Account
3	189	12,368.90	-3,196.56	386.94
4	188	13,922.07	-3,293.10	422.76
5	187	13,570.41	-3,293.10	412.09
6	187	14,670.83	-3,291.60	445.70
7	186	14,410.98	-3,541.80	406.88
8	186	13,960.58	-3,541.80	394.17
9	186	13,561.73	-3,371.10	402.29
10	186	13,578.25	-3,371.10	402.78
11	185	13,569.71	-3,531.08	384.29
12	184	13,683.81	-3,531.08	387.52
13	183	13,970.10	-3,531.08	395.63
14	182	14,240.85	-3,334.94	427.02
15	183	14,021.86	-3,334.94	420.45
16	183	13,925.53	-3,398.00	409.82
17	183	13,334.46	-3,398.00	392.42
18	183	13,393.74	-3,625.04	369.48
19	183	13,478.34	-3,625.04	371.81
20	182	13,392.16	-3,528.95	379.49
21	182	13,213.58	-3,462.18	381.66
22	181	14,227.99	-3,462.18	410.95
23	181	14,043.76	-3,439.40	408.32
24	181	13,882.53	-3,490.83	397.69
25	181	14,609.38	-3,644.97	400.81
26	182	14,616.30	-3,644.97	401.00
27	183	15,799.68	-3,697.66	427.29
28	182	15,488.06	-3,834.96	403.86
29	183	15,729.00	-3,701.30	424.96
30	183	15,098.18	-3,763.38	401.19
31	183	15,416.12	-4,005.40	384.88
32	183	15,155.29	-4,005.40	378.37
33	183	15,089.90	-3,688.44	409.11
34	183	15,519.67	-3,898.86	398.06
35	182	15,351.44	-3,898.86	393.74
36	182	15,684.31	-3,727.40	420.78
37	182	15,622.92	-3,837.78	407.08
38	182	15,742.17	-3,876.00	406.14
39	181	15,784.40	-3,876.00	407.23
40	180	15,775.18	-3,655.32	431.57
41	181	15,148.88	-3,655.32	414.43
42	180	15,595.11	-3,371.10	462.61
43	180	15,020.85	-3,531.08	425.39
44	179	13,556.18	-3,334.94	406.49
45	179	13,760.33	-3,398.00	404.95
46	179	13,369.26	-3,398.00	393.45
47	179	13,172.49	-3,686.89	357.28
48	179	12,872.31	-4,043.06	318.38
49	180	13,007.25	-4,043.06	321.72
50	180	12,965.62	-4,093.81	316.71

Figure 4.3 – FastLen optimization report

In *Figure 4.3*, you can see 48 rows (from rows 3 to 50), and each row represents a **`FastLen`** value. There are five columns:

- **`FastLen`**
- **`Total Trades`**
- **`Net Profit`**
- **`Max Intraday Drawdown`**
- **`Return on Account`**

`FastLen` is the setting to be analyzed, while the remaining four columns represent the four key performance indicators we discussed in the previous chapter. The Optimization Report can also be visualized as in the chart in *Figure 4.4*:

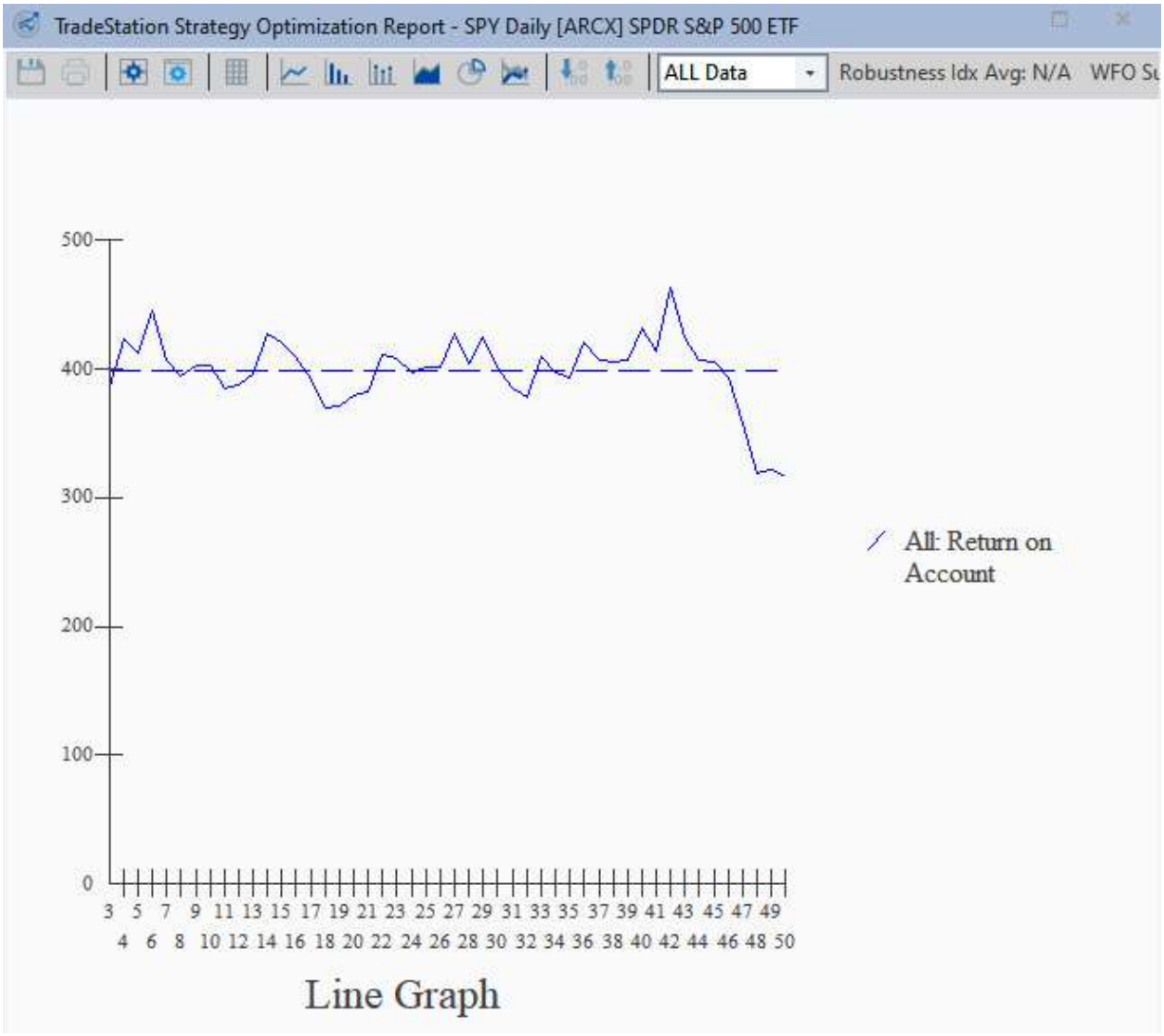

TradeStation Strategy Optimization Report - SPY Daily [ARCX] SPDR S&P 500 ETF
ALL Data
Robustness Idx Avg: N/A
500
400
300
200
100
0
All: Return on Account
3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49
4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50
Line Graph

Figure 4.4 – Return on Account by FastLen

From *Figure 4.4*, we can easily note that the **Return on Account** (**ROA**) is quite stable around 400%, with `FastLen` ranging from `3` to `44`, while above `44` it starts deteriorating.

Let's now vary the `SlowLen` parameters, going through the same optimization process by setting the optimization like *Figure 4.5*:

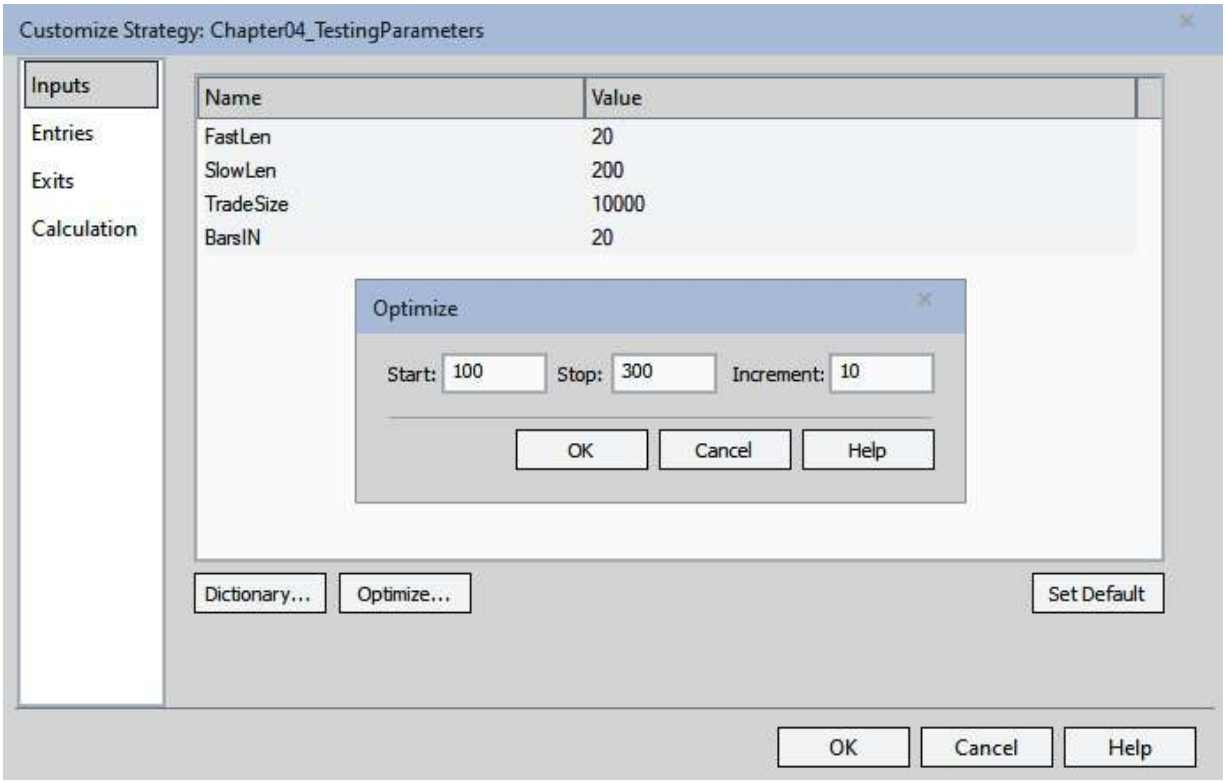


Figure 4.5 – Optimizing SlowLen

In *Figure 4.5*, you can see how to vary the `SlowLen` input from `100` to `300` with an increment of `10`. By running such an optimization, TradeStation will produce the report shown in *Figure 4.6*:

_TestingP SlowLen	All: Net Profit	All: Total Trades	All: Max Intraday Drawdown	All: Return on Account
100	9,690.89	173	-3,416.04	283.69
110	8,790.63	172	-3,614.18	243.23
120	10,830.79	174	-3,531.08	306.73
130	9,126.92	174	-3,398.00	268.60
140	11,690.89	173	-3,625.04	322.50
150	13,242.04	173	-3,528.95	375.24
160	14,198.31	173	-3,528.95	402.34
170	14,771.38	176	-3,528.95	418.58
180	14,722.72	176	-3,528.95	417.20
190	14,431.34	176	-3,528.95	408.94
200	13,570.12	178	-3,528.95	384.54
210	13,936.55	178	-3,625.04	384.45
220	14,187.86	178	-3,625.04	391.38
230	14,025.54	177	-3,398.00	412.76
240	12,039.99	178	-3,398.00	354.33
250	12,265.15	180	-3,398.00	360.95
260	11,564.29	180	-3,398.00	340.33
270	13,181.21	181	-3,398.00	387.91
280	13,514.33	181	-3,398.00	397.71
290	13,628.36	180	-3,398.00	401.07
300	14,545.84	181	-3,398.00	428.07

Figure 4.6 – SlowLen Optimization Report

The Optimization Report in *Figure 4.6* could be visualized in a chart as in *Figure 4.7*.

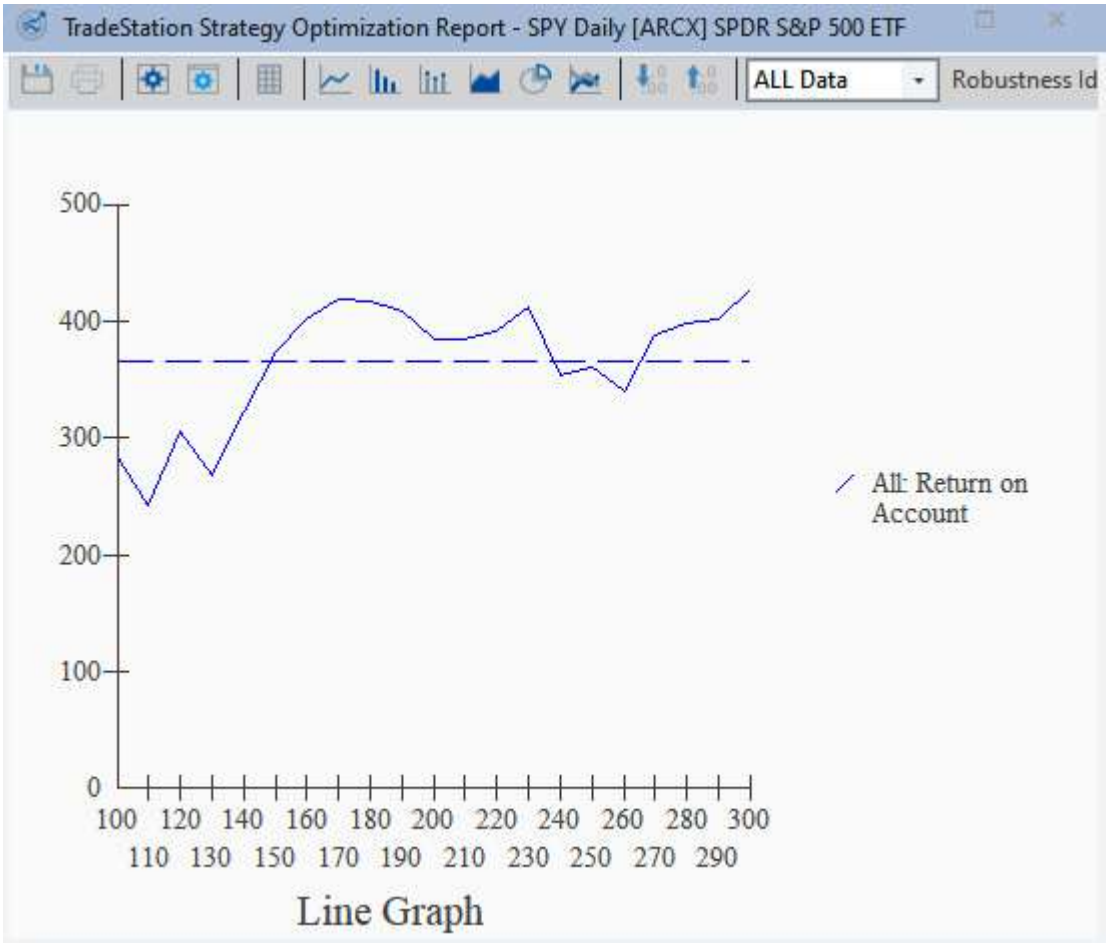


Figure 4.7 – ROA by SlowLen

Analyzing *Figure 4.7*, we can see that varying `SlowLen` from `160` to `230` produces quite a stable **ROA**, above 400%. On the other hand, `SlowLen` values below `140` produce worse results.

A one-way sensitivity analysis is a straightforward tool to evaluate strategy settings: but how do we handle multiple settings? To understand that, we are going to look at a double-way sensitivity analysis.

Double-way sensitivity analysis

In this section, we are going to run a double-way sensitivity analysis by using two different approaches:

- TradeStation Optimization Report
- TradeStation and Excel Optimization Report

TradeStation Optimization Report

Let’s start a new optimization by varying **`FastLen`** from **`3`** to **`50`** in *Step 1* and **`SlowLen`** from **`100`** to **`300`** in *Step 10*.

In *Figure 4.8*, you can see how to set the optimization inputs:

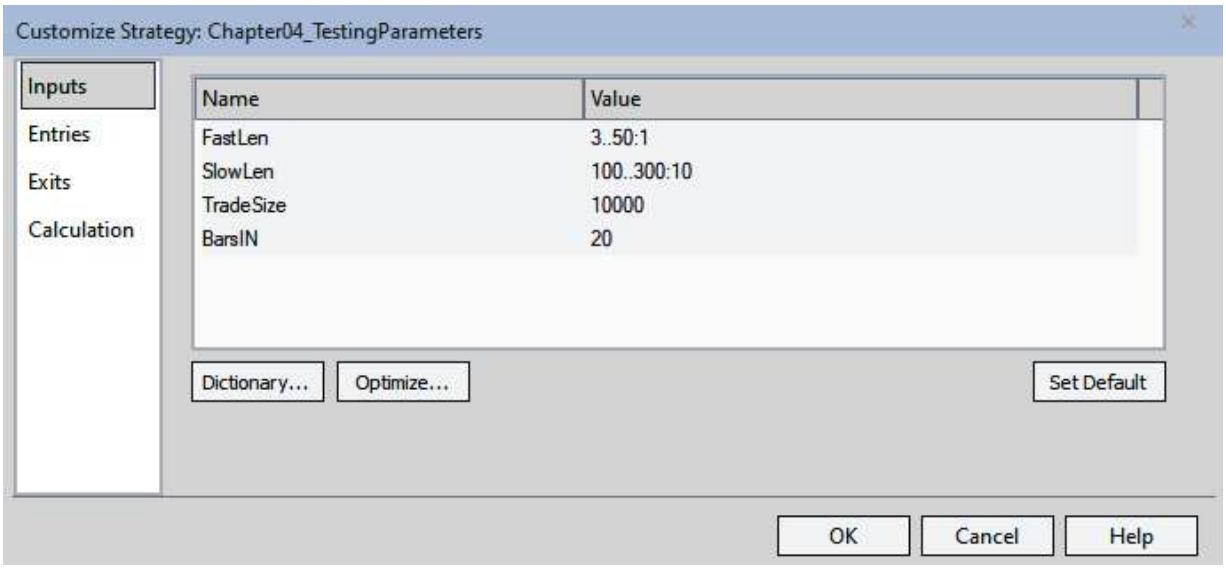


Figure 4.8 – Double-way optimization

By running the optimization in *Figure 4.8*, TradeStation will produce a 1,008-row Optimization Report, as in *Figure 4.9*:

	_TestingP FastLen	_TestingPa SlowLen	All: Net Profit	All: Total Trades	All: Max Intraday Drawdown	All: Return on Account
1	43	220	17,654.63	176	-3,371.10	523.71
2	42	210	16,942.04	176	-3,371.10	502.57
3	49	140	16,925.96	173	-3,371.10	502.09
4	44	160	16,774.41	176	-3,371.10	497.59
5	48	140	16,575.39	173	-3,371.10	491.69
6	43	160	16,565.25	175	-3,371.10	491.39
7	46	150	16,514.95	175	-3,371.10	489.90
8	41	140	18,157.50	173	-3,727.40	487.14
9	42	220	17,670.25	177	-3,655.32	483.41
10	41	220	17,652.46	177	-3,655.32	482.93
11	42	200	16,205.38	176	-3,371.10	480.71
12	9	300	15,756.29	181	-3,293.10	478.46
13	45	220	16,934.53	175	-3,547.49	477.37
14	6	240	15,685.96	181	-3,293.10	476.33
15	12	300	15,565.32	182	-3,291.60	472.88
16	44	220	16,660.34	175	-3,531.08	471.82
17	37	160	17,573.42	174	-3,727.40	471.47
18	40	210	17,196.79	175	-3,655.32	470.46
19	11	300	15,482.43	182	-3,291.60	470.36
20	5	240	15,466.52	181	-3,293.10	469.66
21	41	210	17,151.40	176	-3,655.32	469.22
22	5	250	15,424.63	181	-3,293.10	468.39
23	8	300	15,407.58	182	-3,293.10	467.87
24	42	160	17,097.71	175	-3,655.32	467.75
25	6	200	15,310.70	183	-3,291.60	465.14
26	42	140	17,332.04	173	-3,727.40	464.99
27	34	220	17,139.51	176	-3,688.44	464.68
28	42	190	15,654.32	176	-3,371.10	464.37
29	41	160	16,966.67	174	-3,655.32	464.16
30	42	180	15,621.92	176	-3,371.10	463.41

Figure 4.9 – Double-way Optimization Report

In *Figure 4.9*, you can see the first 30 combinations of **`FastLen`** and **`SlowLen`** sorted by ROA. For example, at the top of the list, we find the **`FastLen=43`** and **`SlowLen=220`** pair, showing the highest ROA (523.71%) among the 1,008 combinations analyzed.

In order to have a better double-way Optimization Report, we could use TradeStation and Excel together. Let me show you how.

TradeStation and Excel Optimization Report

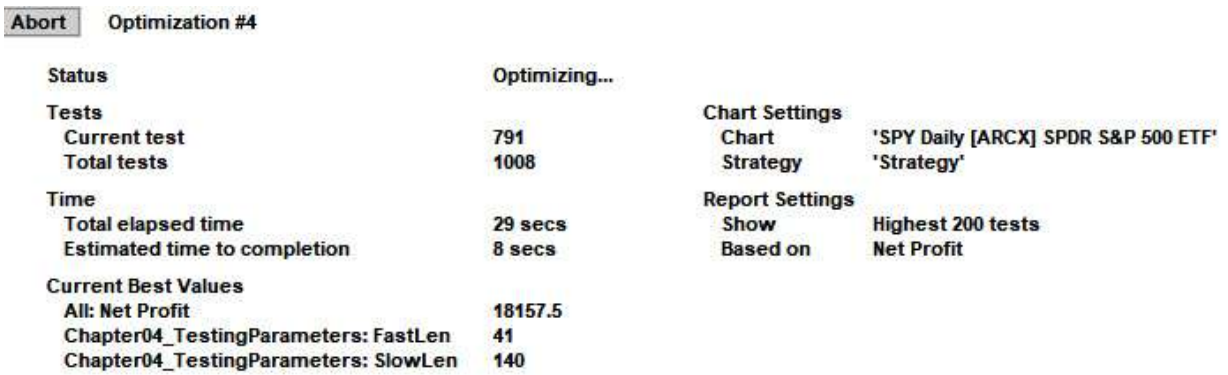
The process of developing trading ideas often involves teamwork; therefore, it is crucial to easily share comprehensive reports. I find it very useful to work with TradeStation and Excel together, as I can enhance the reporting aspect by leveraging Excel's flexibility.

In order to achieve this task, we need to add the following lines at the bottom of our strategy's script:

```
var:ROA(0);
if totaltrades>0 then ROA = -NetProfit/Maxiddrawdown;
if lastbaronchart then print(
symbol,",",
FastLen,",",
SlowLen,",",
Totaltrades,",",
NetProfit,",",
Maxiddrawdown,",",
ROA)
```

The preceding lines will print comma-separated strings, as in *Figure 4.10*, for each backtesting run.

So, running the double-way optimization, the TradeStation Print Log will be populated with data ready to be copied into Excel.



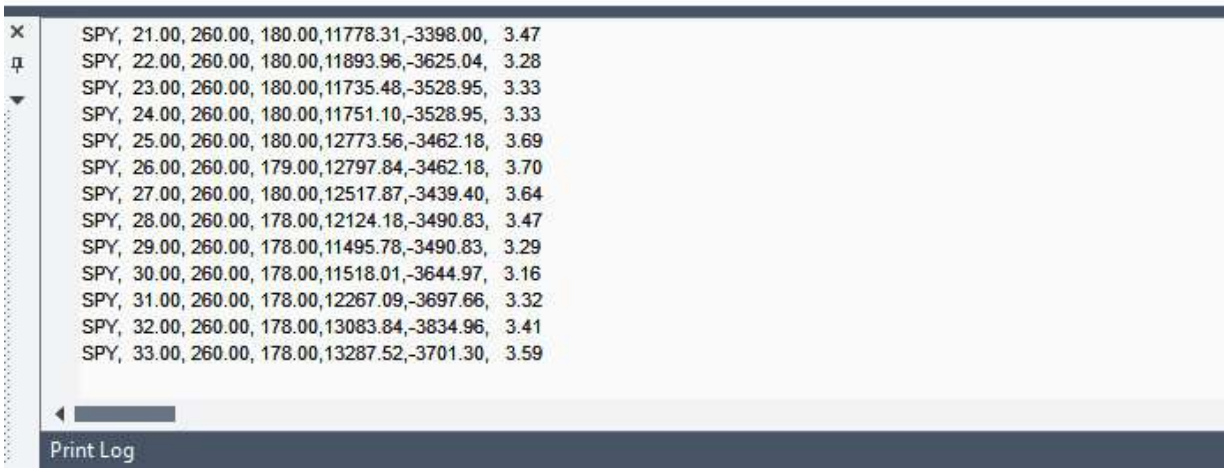


Figure 4.10 – Tradestation Print Log with key performance indicators for FastLen-SlowLen pairs

In *Figure 4.10*, you can see the TradeStation Print Log populated with 1,008 comma-separated strings. Here are the steps to import the comma-separated strings into Excel:

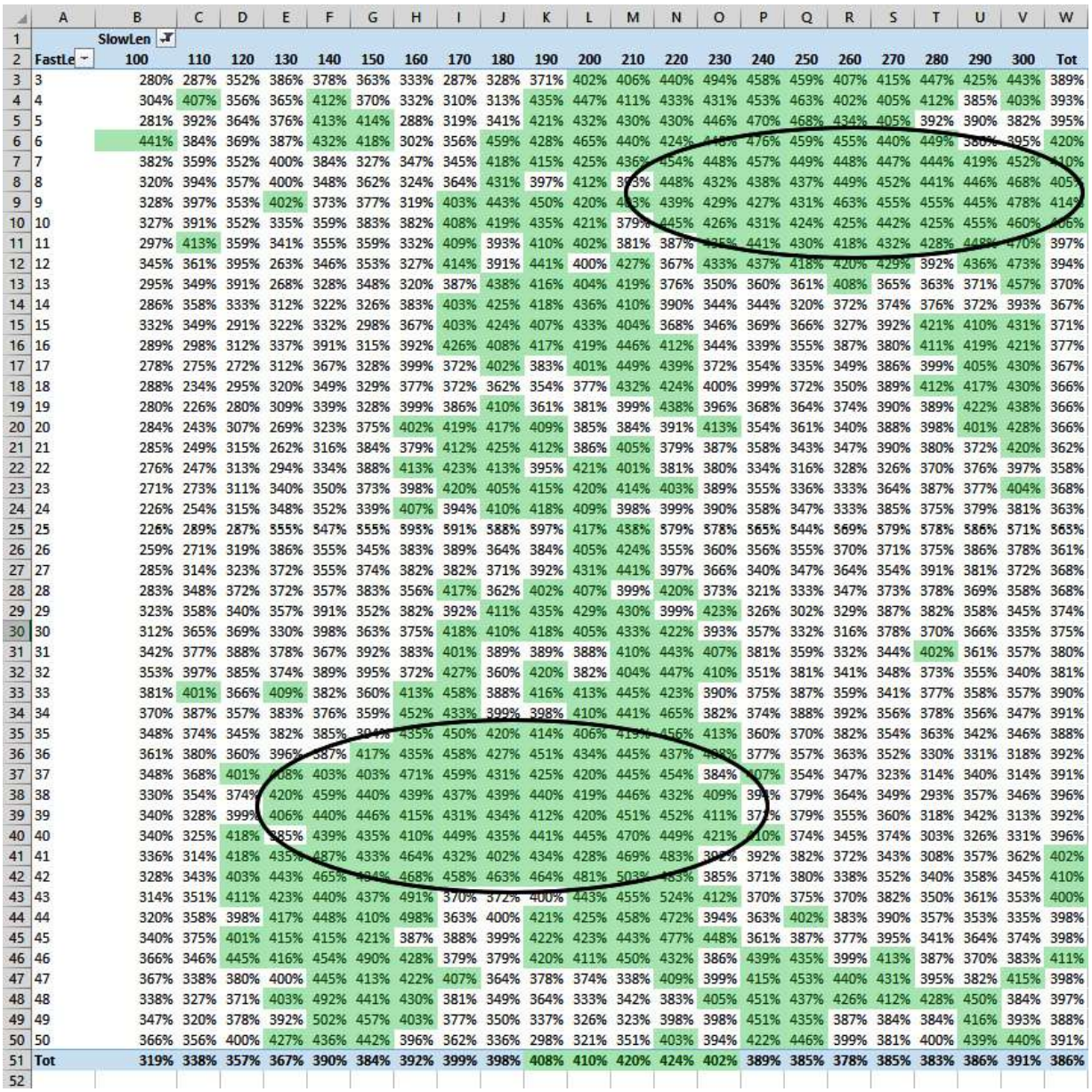
1. Open Excel on your computer.
2. Paste the data you copied from the TradeStation Print Log.
3. Go to **Data** | **Text in column** | **Delimited** | **Comma**.
4. Review the data preview to ensure it looks correct, then click **Finish**.
5. Your comma-separated string data will now be set into Excel, with each value separated into individual cells in the selected range.

By copying and pasting this dataset into an empty Excel file, you can visualize the data as in *Figure 4.11*.

	A	B	C	D	E	F	G
1	Symbol	FastLen	SlowLen	TotalTrades	NetProfit	MaxIdDrawDown	ROA
2	SPY	3	100	179	10093.24	-3603.49	2.8
3	SPY	4	100	179	10958.96	-3603.49	3.04
4	SPY	7	100	177	11595.97	-3036.12	3.82
5	SPY	5	100	178	10561.11	-3755.06	2.81
6	SPY	8	100	175	10228.61	-3196.56	3.2
7	SPY	6	100	176	11731.11	-2661.72	4.41
8	SPY	9	100	175	10470.69	-3196.56	3.28
9	SPY	10	100	175	10782.12	-3293.1	3.27
10	SPY	11	100	175	9786.97	-3293.1	2.97
11	SPY	12	100	174	10913.88	-3163.8	3.45
12	SPY	14	100	173	9331.56	-3259.42	2.86
13	SPY	13	100	174	9495.29	-3216.68	2.95
14	SPY	15	100	174	10719.27	-3227.42	3.32
15	SPY	16	100	174	9935.36	-3439.8	2.89
16	SPY	18	100	173	9825.32	-3416.04	2.88
17	SPY	17	100	174	9565.2	-3439.8	2.78
18	SPY	19	100	173	9568.72	-3416.04	2.8
19	SPY	20	100	173	9690.89	-3416.04	2.84
20	SPY	21	100	173	10021.22	-3511.08	2.85
21	SPY	22	100	174	9689.43	-3511.08	2.76
22	SPY	24	100	173	8923.22	-3946.28	2.26
23	SPY	23	100	173	9514.75	-3511.08	2.71
24	SPY	25	100	173	8929.04	-3946.28	2.26
25	SPY	26	100	173	9374.97	-3614.18	2.59
26	SPY	27	100	172	10312.27	-3614.18	2.85
27	SPY	28	100	171	9994.03	-3531.08	2.83
28	SPY	29	100	172	11397.23	-3531.08	3.23

Figure 4.11 – Excel populated with the Print Log dataset

Starting from the sheet in *Figure 4.11* and working with an Excel pivot table and conditional formatting, we can produce the heatmap shown in *Figure 4.12*:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1		SlowLen																					
2	FastLe	100	110	120	130	140	150	160	170	180	190	200	210	220	230	240	250	260	270	280	290	300	Tot
3	3	280%	287%	352%	386%	378%	363%	333%	287%	328%	371%	402%	406%	440%	494%	458%	459%	407%	415%	447%	425%	443%	389%
4	4	304%	407%	356%	365%	412%	370%	332%	310%	313%	435%	447%	411%	433%	431%	453%	463%	402%	405%	412%	385%	403%	393%
5	5	281%	392%	364%	376%	413%	414%	288%	319%	341%	421%	432%	430%	430%	446%	470%	468%	434%	405%	392%	390%	382%	395%
6	6	441%	384%	369%	387%	432%	418%	302%	356%	459%	428%	465%	440%	424%	448%	476%	459%	455%	440%	449%	380%	395%	420%
7	7	382%	359%	352%	400%	384%	327%	347%	345%	418%	415%	425%	436%	454%	448%	457%	449%	448%	447%	444%	419%	452%	410%
8	8	320%	394%	357%	400%	348%	362%	324%	364%	431%	397%	412%	393%	448%	432%	438%	437%	449%	452%	441%	446%	468%	405%
9	9	328%	397%	353%	402%	373%	377%	319%	403%	443%	450%	420%	403%	439%	429%	427%	431%	463%	455%	455%	445%	478%	414%
10	10	327%	391%	352%	335%	359%	383%	382%	408%	419%	435%	421%	379%	445%	426%	431%	424%	425%	442%	425%	455%	460%	406%
11	11	297%	413%	359%	341%	355%	359%	332%	409%	393%	410%	402%	381%	387%	435%	441%	430%	418%	432%	428%	448%	470%	397%
12	12	345%	361%	395%	263%	346%	353%	327%	414%	391%	441%	400%	427%	367%	433%	437%	418%	420%	429%	392%	436%	473%	394%
13	13	295%	349%	391%	268%	328%	348%	320%	387%	438%	416%	404%	419%	376%	350%	360%	361%	408%	365%	363%	371%	457%	370%
14	14	286%	358%	333%	312%	322%	326%	383%	403%	425%	418%	436%	410%	390%	344%	344%	320%	372%	374%	376%	372%	393%	367%
15	15	332%	349%	291%	322%	332%	298%	367%	403%	424%	407%	433%	404%	368%	346%	369%	366%	327%	392%	421%	410%	431%	371%
16	16	289%	298%	312%	337%	391%	315%	392%	426%	408%	417%	419%	446%	412%	344%	339%	355%	387%	380%	411%	419%	421%	377%
17	17	278%	275%	272%	312%	367%	328%	399%	372%	402%	383%	401%	449%	439%	372%	354%	335%	349%	386%	399%	405%	430%	367%
18	18	288%	234%	295%	320%	349%	329%	377%	372%	362%	354%	377%	432%	424%	400%	399%	372%	350%	389%	412%	417%	430%	366%
19	19	280%	226%	280%	309%	339%	328%	399%	386%	410%	361%	381%	399%	438%	396%	368%	364%	374%	390%	389%	422%	438%	366%
20	20	284%	243%	307%	269%	323%	375%	402%	419%	417%	409%	385%	384%	391%	413%	354%	361%	340%	388%	398%	401%	428%	366%
21	21	285%	249%	315%	262%	316%	384%	379%	412%	425%	412%	386%	405%	379%	387%	358%	343%	347%	390%	380%	372%	420%	362%
22	22	276%	247%	313%	294%	334%	388%	413%	423%	413%	395%	421%	401%	381%	380%	334%	316%	328%	326%	370%	376%	397%	358%
23	23	271%	273%	311%	340%	350%	373%	398%	420%	405%	415%	420%	414%	403%	389%	355%	336%	333%	364%	387%	377%	404%	368%
24	24	226%	254%	315%	348%	352%	339%	407%	394%	410%	418%	409%	398%	399%	390%	358%	347%	333%	385%	375%	379%	381%	363%
25	25	226%	289%	287%	355%	347%	355%	393%	391%	388%	397%	417%	438%	379%	378%	365%	344%	369%	379%	378%	386%	371%	363%
26	26	259%	271%	319%	386%	355%	345%	383%	389%	364%	384%	405%	424%	355%	360%	356%	355%	370%	371%	375%	386%	378%	361%
27	27	285%	314%	323%	372%	355%	374%	382%	382%	371%	392%	431%	441%	397%	366%	340%	347%	364%	354%	391%	381%	372%	368%
28	28	283%	348%	372%	372%	357%	383%	356%	417%	362%	402%	407%	399%	420%	373%	321%	333%	347%	373%	378%	369%	358%	368%
29	29	323%	358%	340%	357%	391%	352%	382%	392%	411%	435%	429%	430%	399%	423%	326%	302%	329%	387%	382%	358%	345%	374%
30	30	312%	365%	369%	330%	398%	363%	375%	418%	410%	418%	405%	433%	422%	393%	357%	332%	316%	378%	370%	366%	335%	375%
31	31	342%	377%	388%	378%	367%	392%	383%	401%	389%	389%	388%	410%	443%	407%	381%	359%	332%	344%	402%	361%	357%	380%
32	32	353%	397%	385%	374%	389%	395%	372%	427%	360%	420%	382%	404%	447%	410%	351%	381%	341%	348%	373%	355%	340%	381%
33	33	381%	401%	366%	409%	382%	360%	413%	458%	388%	416%	413%	445%	423%	390%	375%	387%	359%	341%	377%	358%	357%	390%
34	34	370%	387%	357%	383%	376%	359%	452%	433%	399%	398%	410%	441%	465%	382%	374%	388%	392%	356%	378%	356%	347%	391%
35	35	348%	374%	345%	382%	385%	394%	435%	450%	420%	414%	406%	419%	456%	413%	360%	370%	382%	354%	363%	342%	346%	388%
36	36	361%	380%	360%	396%	387%	417%	435%	458%	427%	451%	434%	445%	437%	408%	377%	357%	363%	352%	330%	331%	318%	392%
37	37	348%	368%	401%	408%	403%	403%	471%	459%	431%	425%	420%	445%	454%	384%	407%	354%	347%	323%	314%	340%	314%	391%
38	38	330%	354%	374%	420%	459%	440%	439%	437%	439%	440%	419%	446%	432%	409%	393%	379%	364%	349%	293%	357%	346%	396%
39	39	340%	328%	399%	406%	440%	446%	415%	431%	434%	412%	420%	451%	452%	411%	378%	379%	355%	360%	318%	342%	313%	392%
40	40	340%	325%	418%	385%	439%	435%	410%	449%	435%	441%	445%	470%	449%	421%	410%	374%	345%	374%	303%	326%	331%	396%
41	41	336%	314%	418%	435%	487%	433%	464%	432%	402%	434%	428%	469%	483%	392%	392%	382%	372%	343%	308%	357%	362%	402%
42	42	328%	343%	403%	443%	465%	434%	468%	458%	463%	464%	481%	503%	483%	385%	371%	380%	338%	352%	340%	358%	345%	410%
43	43	314%	351%	411%	423%	440%	437%	491%	370%	372%	400%	443%	455%	524%	412%	370%	375%	370%	382%	350%	361%	353%	400%
44	44	320%	358%	398%	417%	448%	410%	498%	363%	400%	421%	425%	458%	472%	394%	363%	402%	383%	390%	357%	353%	335%	398%
45	45	340%	375%	401%	415%	415%	421%	387%	388%	399%	422%	423%	443%	477%	448%	361%	387%	377%	395%	341%	364%	374%	398%
46	46	366%	346%	445%	416%	454%	490%	428%	379%	379%	420%	411%	450%	432%	386%	439%	435%	399%	413%	387%	370%	383%	411%
47	47	367%	338%	380%	400%	445%	413%	422%	407%	364%	378%	374%	338%	409%	399%	415%	453%	440%	431%	395%	382%	415%	398%
48	48	338%	327%	371%	403%	492%	441%	430%	381%	349%	364%	333%	342%	383%	405%	451%	437%	426%	412%	428%	450%	384%	397%
49	49	347%	320%	378%	392%	502%	457%	403%	377%	350%	337%	326%	323%	398%	398%	451%	435%	387%	384%	384%	416%	393%	388%
50	50	366%	356%	400%	427%	436%	442%	396%	362%	336%	298%	321%	351%	403%	394%	422%	446%	399%	381%	400%	439%	440%	391%
51	Tot	319%	338%	357%	367%	390%	384%	392%	399%	398%	408%	410%	420%	424%	402%	389%	385%	378%	385%	383%	386%	391%	386%
52																							

Figure 4.12 – Excel heatmap reporting the ROA (Return on Asset) for any combination of FastLen/SlowLen

If you look at the heatmap in *Figure 4.12*, you can see a pivot table where you have the **`FastLen`** settings in column A and the **`SlowLen`** settings in row 2.

Any cell in this pivot table reports the ROA for the corresponding pair of **`FastLen`** and **`SlowLen`**; for example, cell B3 reports an ROA of 280%, which comes from a backtesting performed with a **`FastLen`** value of **`3`** and a **`SlowLen`** value of **`100`**.

I have conditionally colored the pivot table cells with an ROA above 400% in order to have a clear picture of how the ROA changes according to the two features analyzed. As you can see in the circles, there are two areas with a stable ROA above 400%:

- The area with **FastLen** from **6** to **11** matched with **SlowLen** being between **220** and **300**
- The area with **FastLen** from **35** to **42** matched with **SlowLen** being from **130** to **220**

In plain English, this analysis tells us two things:

- We could try to catch a 1/1.5-year trend (**SlowLen** between 220 and 300 days) when a weekly trend is in place (**FastLen** between 6 and 11 days)
- We could catch a 0.5/1-year trend (**SlowLen** between 130 and 220 days) when a 2-month trend is in place (**FastLen** between 35 and 45 days)

Using colors is very helpful when you need to analyze a huge amount of data.

Multiple-way sensitivity analysis

In the case of strategies with more than two features, the best approach I have found to avoid overfitting is to conduct a double-way sensitivity analysis multiple times. For example, going back to *Figure 4.12*, you can run the same optimization with a **BarsIN** value of 40 days instead of 20 days and then compare the results.

In this section, we learned how to handle sensitivity analysis, that is, backtesting different parameters for the same financial asset. But what will happen if we apply the same strategy to a different asset?

Backtesting across symbols

To check whether the strategy settings have fit too much to a single asset of historical data, it is a good habit to test the same strategies across similar assets. In this section, we are going to do the following:

- Understand the TradeStation stock symbol universe
- Produce a ROA report in Excel
- Produce an equity line chart in Excel

TradeStation stock symbol universe

The TradeStation stock symbol universe, as of February 2024, is summarized in *Table 4.1*:

Exchange	**Tickers Available**
NYSE	3033
NASDAQ	3200

Exchange	**Tickers Available**
AMEX	355
ARCX	2110

Table 4.1 – TradeStation stock symbol universe

As you can see in *Table 4.1*, there are more than 11,000 symbols available. They can be grouped into several lists, as in *Figure 4.13*:

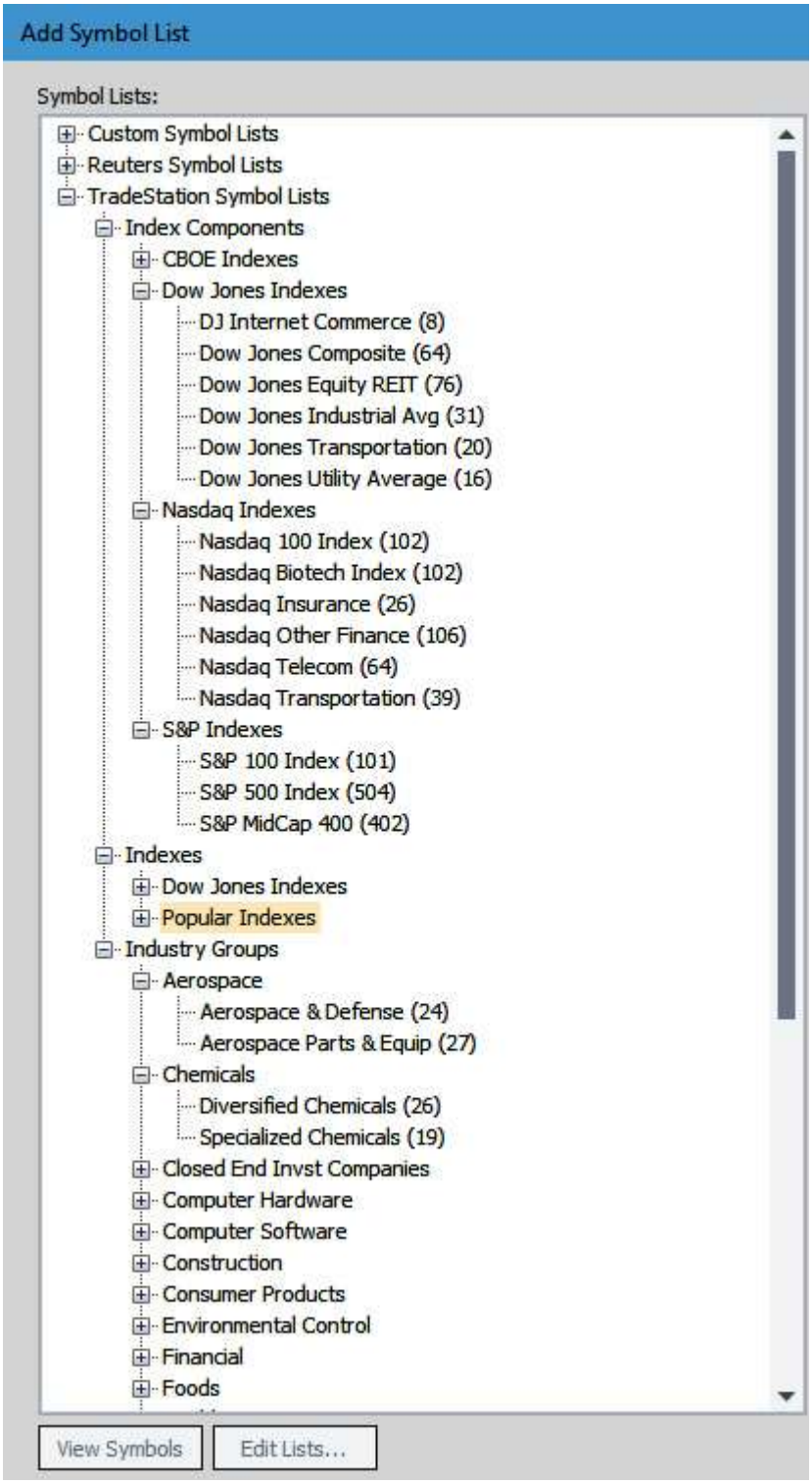


Figure 4.13 – TradeStation Symbol Lists

You can choose, among the many lists available in TradeStation (*Figure 4.13*), the ones you are mainly interested in. For this chapter, we selected Dow Jones 30, which is a list of 30 large, publicly owned companies trading on the **New York Stock Exchange** (**NYSE**) and NASDAQ.

ROA report in Excel

To perform multiple backtests on the Dow Jones 30 stocks, we have to select one strategy set. When choosing strategy settings within an optimization, it's a good practice to choose parameters with positive neighborhoods to avoid random results. This is where a heatmap can help. For example, from the report in *Figure 4.12*, we can select the **`FastLen=10`** and **`SlowLen=260`** pair as it is right in the middle of the top-right circle showing a stable ROA above 400%.

To produce a ROA comparison report in Excel, let's follow these instructions:

1. Arrange a TradeStation workspace with an empty RadarScreen and a chart with the *Fast SMA above Slow SMA* strategy set with **`FastLen=10`** and **`SlowLen=260`**.
2. Link RadarScreen to the chart by assigning the same color in the small **S** button on top of both Chart and RadarScreen.
3. From the main menu, go to **View** | **EasyLanguage Print Log**.

   A Print Log window will appear at the bottom of your screen.
4. Clear the Print Log (a right-click on the window will open a menu with the clear option).
5. From the RadarScreen menu, go to **Data** | **Add Symbol List** and select **Dow Jones Industrial**.
6. From top to bottom, click on any symbol in the RadaScreen's list.
7. The chart will change accordingly, the strategy will be re-calculated based on the new symbol selected, and the Print Log will be updated with a new line (*Figure 4.14*).

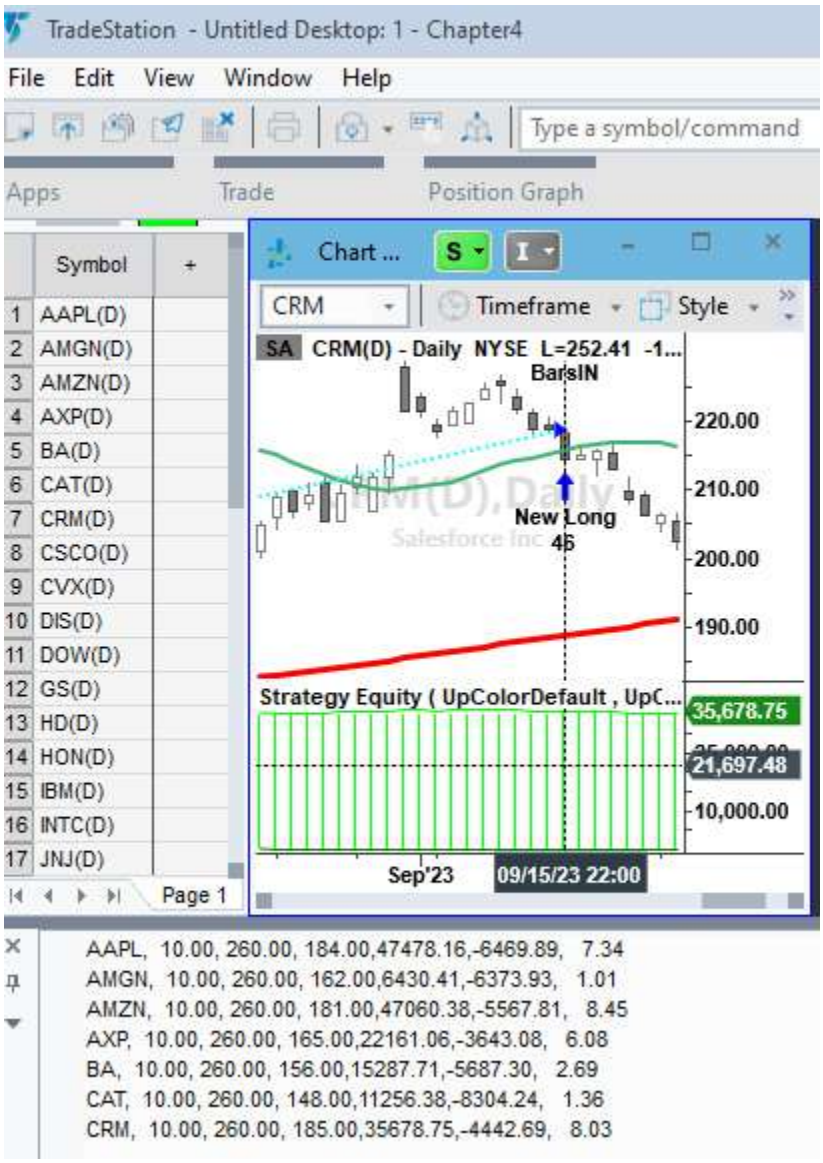
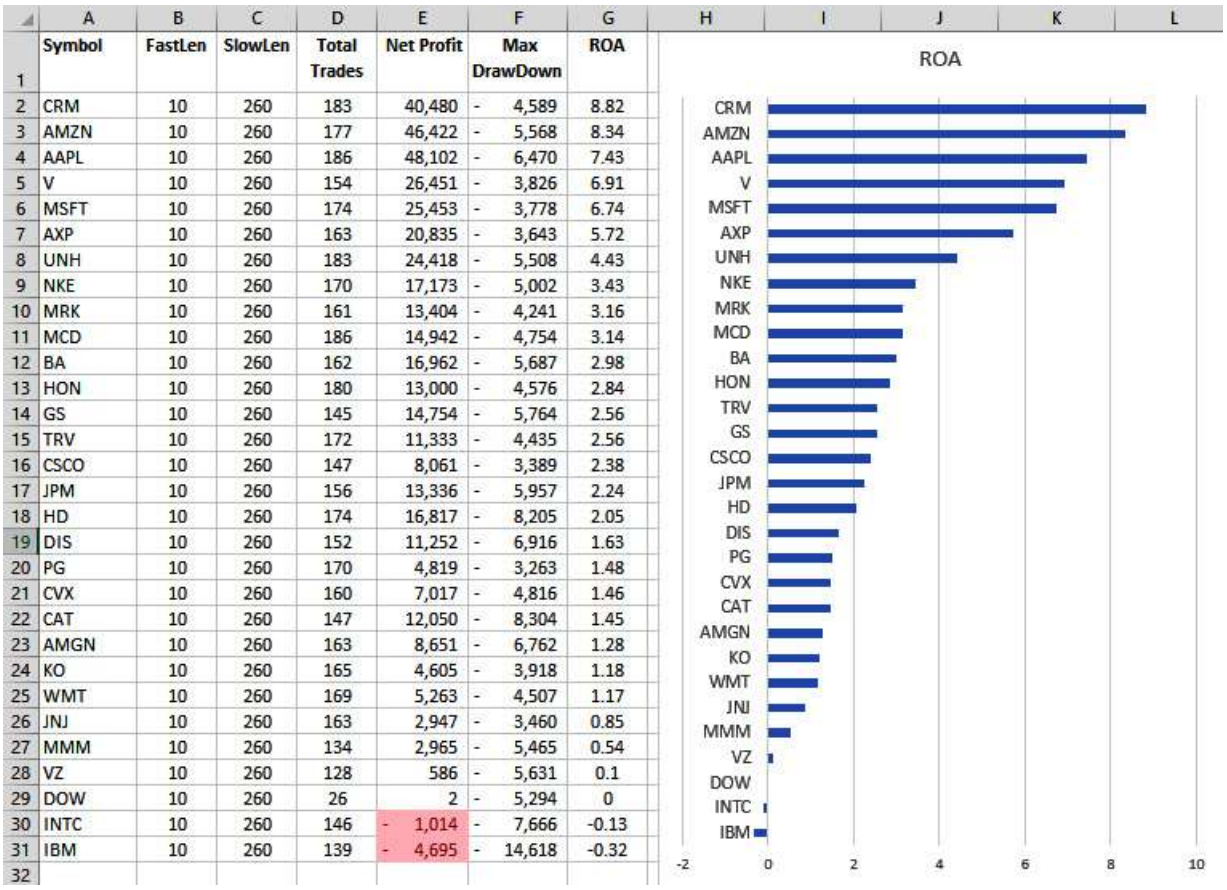


Figure 4.14 – Backtesting on the Dow Jones 30 list using the Print Log

Once you have repeated the same process for the 30 symbols, you will have 30 lines in the Print Log (*Figure 4.14*).

Let's now copy and paste the Print Log content into an empty Excel file (*Figure 4.15*). You can manage the Print Log data using *Ctrl + A* to select all the lines and *Ctrl + C* to copy the content.



	Symbol	FastLen	SlowLen	Total Trades	Net Profit	Max DrawDown	ROA
2	CRM	10	260	183	40,480	- 4,589	8.82
3	AMZN	10	260	177	46,422	- 5,568	8.34
4	AAPL	10	260	186	48,102	- 6,470	7.43
5	V	10	260	154	26,451	- 3,826	6.91
6	MSFT	10	260	174	25,453	- 3,778	6.74
7	AXP	10	260	163	20,835	- 3,643	5.72
8	UNH	10	260	183	24,418	- 5,508	4.43
9	NKE	10	260	170	17,173	- 5,002	3.43
10	MRK	10	260	161	13,404	- 4,241	3.16
11	MCD	10	260	186	14,942	- 4,754	3.14
12	BA	10	260	162	16,962	- 5,687	2.98
13	HON	10	260	180	13,000	- 4,576	2.84
14	GS	10	260	145	14,754	- 5,764	2.56
15	TRV	10	260	172	11,333	- 4,435	2.56
16	CSCO	10	260	147	8,061	- 3,389	2.38
17	JPM	10	260	156	13,336	- 5,957	2.24
18	HD	10	260	174	16,817	- 8,205	2.05
19	DIS	10	260	152	11,252	- 6,916	1.63
20	PG	10	260	170	4,819	- 3,263	1.48
21	CVX	10	260	160	7,017	- 4,816	1.46
22	CAT	10	260	147	12,050	- 8,304	1.45
23	AMGN	10	260	163	8,651	- 6,762	1.28
24	KO	10	260	165	4,605	- 3,918	1.18
25	WMT	10	260	169	5,263	- 4,507	1.17
26	JNJ	10	260	163	2,947	- 3,460	0.85
27	MMM	10	260	134	2,965	- 5,465	0.54
28	VZ	10	260	128	586	- 5,631	0.1
29	DOW	10	260	26	2	- 5,294	0
30	INTC	10	260	146	- 1,014	- 7,666	-0.13
31	IBM	10	260	139	- 4,695	- 14,618	-0.32

Figure 4.15 – Excel file to elaborate on backtesting across the Dow Jones 30 symbol list

In *Figure 4.15*, you can see an example of how you can use Excel to produce fully customizable reports. From this report, we can see that the strategy produced a positive performance on 28 out of 30 stocks. In this example, I have added a chart displaying the ROA by asset.

> ***NOTE***
>
> *You can customize the Excel file according to your analytical needs. For example, you can add a column with the stock's sector and then create a new pivot table to summarize the ROA by sector. You can also modify the Excel file for presentation purposes, as it is considerably more flexible than a TradeStation report.*

Equity lines chart on Excel

At this point of our across-symbol backtesting, we should be curious about what the equity lines look like. To compare the equity lines of the Dow Jones 30 components, we need to do the following:

1. Modify the print instructions.
2. Re-run the backtestings.

Modifying the print instruction

We need to add the following line at the bottom of our strategy scripts:

```
print( symbol,",",Date,",",NetProfit+Openpositionprofit);
```

This code line will print the following data for any bar on the chart:

- **`Symbol`**: This is the TradeStation reserved word used to refer to the stock's ticker.
- **`Date`**: This is any date, in EasyLanguage format, starting from the left of the chart and ending to the right of it
- **`NetProfit+Openpositionprofit`**: This is the sum of the closed profit (**`NetProfit`**) and the open profit (**`Openpositionprofit`**)

> ***NOTE***
>
> *In EasyLanguage, the date format typically follows the YYYMMDD structure, for example, **`1240628`** = June 6th, 2024.*

Re-running the backtesting

We now need to re-run the backtesting by following this procedure:

1. Clear re-running" the Print Log (by right-clicking on the window, a menu will be opened).
2. From top to bottom of RadarScreen, click on any symbol in the list.
3. The chart will change accordingly, the strategy will be re-calculated on the new symbol selected, and the Print Log will be updated with a new line.
4. Once done with all 30 rows, copy the Print Log content by using the *Ctrl + A* keys and paste it into an empty Excel.

The Excel file will have 3 columns and around 140,000 rows (*Figure 4.16*):

	A	B	C
1	symbol	date	Profit
2	AAPL	1050330	0
3	AAPL	1050331	-196.08
4	AAPL	1050401	-392.16
5	AAPL	1050404	-326.8
6	AAPL	1050405	-130.72
7	AAPL	1050406	-65.36
8	AAPL	1050407	261.44
9	AAPL	1050408	261.44
10	AAPL	1050411	-130.72
11	AAPL	1050412	0
12	AAPL	1050413	-326.8
13	AAPL	1050414	-1241.84
14	AAPL	1050415	-1699.36
15	AAPL	1050418	-1634
16	AAPL	1050419	-1307.2
17	AAPL	1050420	-1634
18	AAPL	1050421	-1241.84
19	AAPL	1050422	-1634
138356	WMT	1240227	5168.95
138357	WMT	1240228	5174.26
138358	WMT	1240229	4995.49
138359	WMT	1240301	5022.04
138360	WMT	1240304	5117.62
138361	WMT	1240305	5248.6
138362	WMT	1240306	5337.91
138363	WMT	1240307	5302.84
138364	WMT	1240308	5262.76
138365	WMT	1240311	5352.94
138366	WMT	1240312	5478.19
138367	WMT	1240313	5456.48
138368	WMT	1240314	5413.06
138369	WMT	1240315	5356.28

Figure 4.16 – Excel equity line data collection

Let’s re-running” create a new pivot table using the data from cells A1 to C138369 to visualize a dataset with daily profits as in *Figure 4.17*:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1																				
2	Date	AAPL	WMT	AMGN	AMZN	AXP	BA	CAT	CRM	CSCO	CVX	DIS	DOW	GS	HD	HON	IBM	INTC	JNJ	JPM
4760	1240222	47,914	4,971	8,708	46,767	20,585	17,066	12,389	40,561	8,007	7,249	11,064	26	14,463	16,416	12,786	4,482	1,071	2,221	13,293
4761	1240223	47,813	4,980	8,844	46,791	20,613	17,034	12,450	40,531	8,063	7,249	11,075	98	14,478	16,434	12,777	4,401	1,068	2,221	13,346
4762	1240226	47,738	5,171	8,748	46,776	20,733	17,043	12,495	40,797	8,061	7,249	11,068	39	14,456	16,424	12,703	4,487	1,068	2,221	13,311
4763	1240227	47,819	5,169	8,480	46,705	20,784	17,043	12,565	40,766	8,061	7,249	11,232	76	14,472	16,534	12,652	4,447	1,127	2,221	13,316
4764	1240228	47,752	5,174	8,445	46,683	20,787	17,043	12,625	40,775	8,061	7,249	11,360	140	14,533	16,592	12,621	4,424	1,295	2,221	13,368
4765	1240229	47,715	4,995	8,322	46,895	20,855	17,043	12,761	41,092	8,061	7,249	11,432	4	14,426	16,676	12,680	4,438	1,055	2,183	13,460
4766	1240301	47,655	5,022	8,543	46,982	20,866	17,043	12,846	41,374	8,061	7,249	11,467	16	14,401	16,775	12,677	4,267	880	2,228	13,418
4767	1240304	47,405	5,118	8,511	46,944	20,831	17,043	12,907	41,296	8,061	7,249	11,629	208	14,509	16,669	12,706	4,005	473	2,087	13,494
4768	1240305	47,131	5,249	8,418	46,740	20,752	17,043	12,748	40,740	8,061	7,249	11,552	201	14,474	16,619	12,610	4,065	1,030	2,095	13,597
4769	1240306	47,076	5,338	8,416	46,704	20,810	17,043	12,813	40,915	8,061	7,249	11,291	179	14,439	16,593	12,770	3,837	723	2,056	13,651
4770	1240307	47,069	5,303	8,289	46,899	21,039	17,043	12,955	40,868	8,061	7,249	11,302	194	14,410	16,570	12,878	3,817	351	2,027	13,560
4771	1240308	47,164	5,263	8,319	46,812	21,036	17,043	12,923	40,968	8,061	7,249	11,315	134	14,372	16,487	12,783	3,849	839	2,067	13,579
4772	1240311	47,275	5,353	8,374	46,612	20,890	17,043	12,793	40,993	8,061	7,249	11,500	281	14,351	16,439	12,760	4,076	644	2,173	13,583
4773	1240312	47,302	5,478	8,414	46,815	21,003	17,043	12,863	41,015	8,061	7,249	11,514	326	14,403	16,517	12,700	3,750	557	2,267	13,668
4774	1240313	47,186	5,456	8,395	46,884	21,011	17,043	12,996	40,947	8,061	7,249	11,518	378	14,511	16,633	12,731	3,808	1,014	2,165	13,753
4775	1240314	47,289	5,413	8,233	47,013	20,903	17,043	13,008	40,900	8,061	7,249	11,477	262	14,414	16,536	12,562	3,985	1,125	2,048	13,565
4776	1240315	47,268	5,356	8,137	46,757	20,810	17,043	13,165	40,585	8,061	7,249	11,467	211	14,378	16,483	12,629	4,112	1,150	1,984	13,693

Figure 4.17 – Excel pivot table

In *Figure 4.18*, you can see the daily net profit for any stock. To visualize re-running" an equity lines chart, let's create a new chart from the pivot table (*Figure 4.18*):

Figure 4.18 – Excel equity line charts

In *Figure 4.18*, you can see the individual equity line for the Dow Jones 30 components. Carefully observing the chart allows us to draw observations. Take the following:

- A couple re-running" of lines are always in the negative performance area and we should further investigate the reason for this
- 3 stocks out of 30 show overwhelming performances, maybe due to their individual volatility; we should further investigate this as well

Backtesting versus buy-and-hold

The **buy-and-hold** strategy is an investment approach where an investor purchases securities (such as stocks, bonds, or mutual funds) and holds onto them for an extended period, regardless of short-term market fluctuations. The basic idea behind the buy-and-hold strategy is that over the long term, the market tends to rise, and therefore, holding onto investments for a significant period allows investors to benefit from overall market growth.

When evaluating an algorithmic strategy, we have to answer the question, "What if instead of algorithmically trading the specific asset, we just buy and hold it?" Every trading strategy, to be considered effective, must outperform the buy-and-hold approach. This means it should demonstrate better performance metrics, such as higher returns or lower risk, compared to simply holding the stock over the same period. The strategy should justify the additional effort and complexity by delivering superior results.

To perform this task, I created the **EquityEdge Buy&Hold Analyzer**. The EquityEdge Buy&Hold Analyzer is an EasyLanguage add-on designed to automatically generate a report comparing the results of a trading strategy against the buy-and-hold strategy. It can be applied to any strategy component by going to **Strategies** | **Add Strategy** (*Figure 4.19*):

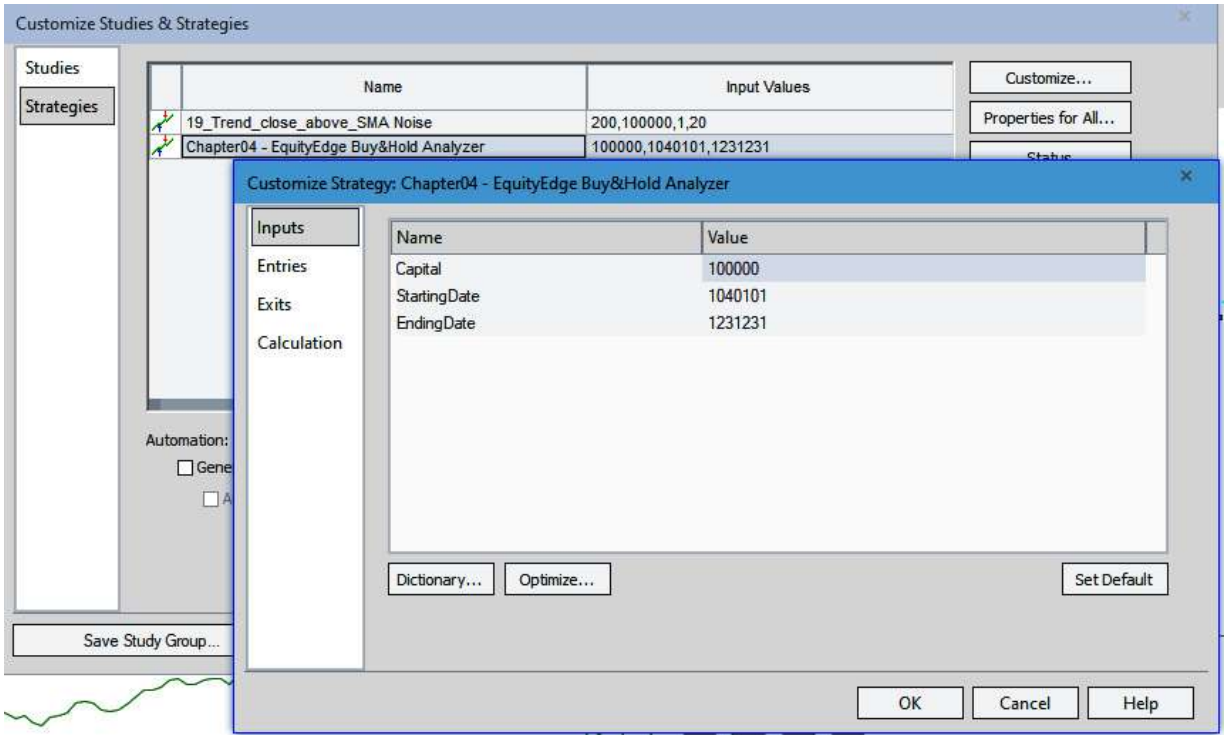


Figure 4.19 – The EquityEdge Buy&Hold Analyzer add-on

In *Figure 4.19*, you can see that EquityEdge Buy&Hold Analyzer requires the following user inputs:

- `Capital`
- `StartingDate`

- **`EndingDate`**

`Capital` represents the trade size in USD. Be sure the trade size set in the Analyzer tool is the same as the strategy.

`StartingDate` and **`EndingDate`** define the period in which the test is carried out.

The Analyzer tool will produce a report like *Figure 4.20* in the TradeStation Print Log:

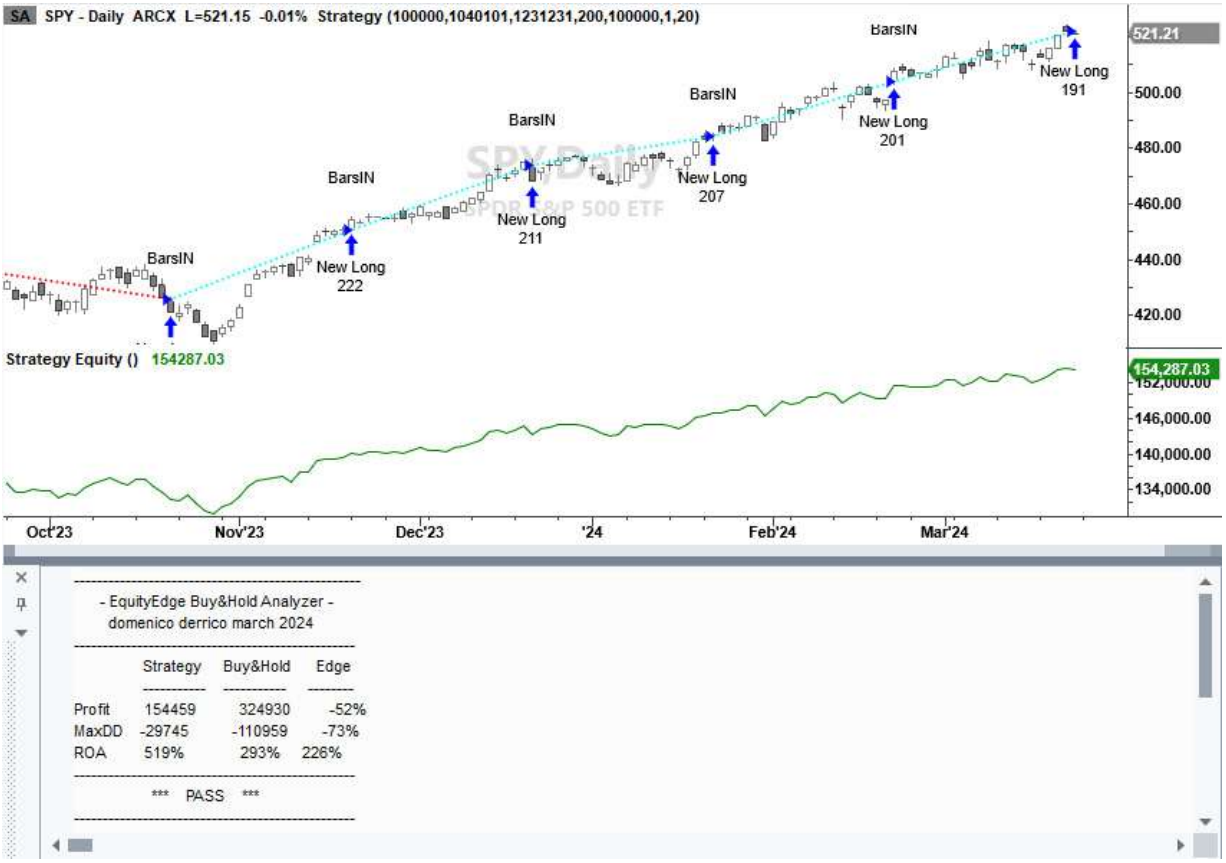


Figure 4.20 – The EquityEdge Buy&Hold Analyzer report

In *Figure 4.20*, you can see the EquityEdge Buy&Hold Analyzer report to measure the trading strategy versus the benchmark. In this example, the strategy's profit drops by 52% versus the buy-and-hold strategy, but the maximum drawdown drops by 73%, meaning that the strategy passed the buy-and-hold test. The EquityEdge Buy&Hold Analyzer script can be found at https://github.com/PacktPublishing/Tradestation-Easylanguage-for-Algorithmic-Trading.

> ***NOTE***
>
> *The EquityEdge Buy&Hold Analyzer script is available on GitHub and is password protected. The password is* ***`PacktChapter04`****.*

In-sample, out-of-sample analysis

The **in-sample, out-of-sample** process consists of using a portion of the available data (the in-sample data) to develop and train a model or conduct analysis. Subsequently, the developed model is tested using a separate portion of the data (known as out-of-sample data) that was not utilized during the training phase. This helps assess the robustness and generalizability of the model or analysis beyond the data it was trained on. The goal is to ensure that the insights or predictions derived from the in-sample data apply to new, unseen data.

In this section, we are going to do the following:

- Modify EasyLanguage scripts for the in-sample, out-of-sample analysis
- Run an example of in-sample, out-of-sample validation

Modifying EasyLanguage scripts for in-sample, out-of-sample purposes

To run an in-sample, out-of-sample validation, you need to modify the EasyLanguage code, as follows:

```
Input:StartDate_(0),EndDate_(1191231);
Input: Len(200),TradeSize(10000);
Var: Qty(0);
Qty = TradeSize/Close;
Input:NoiseFilterType(1);
var:EntryCondition(false);
if NoiseFilterType=0 then EntryCondition = Close >average(Close,Len)[1];
if NoiseFilterType=1 then EntryCondition = Close >average(close,Len)
  and Close[1]>average(Close,Len)[1];
if NoiseFilterType=2 then EntryCondition = Close >average(close,200) +
0.5*StdDev(close,200);
If date>StartDate_ and date<EndDate_ and EntryCondition then Buy ("New Long") Qty Shares
next bar open;
Input:BarsIN(20);
If BarsSinceentry = BarsIN then Sell ("BarsIN") next bar open;
```

Here, the **`If date>StartDate_ and date<EndDate_`** condition restricts the trading to the period between **`StartDate_`** and **`EndDate_`**.

An example of in-sample, out-of-sample validation

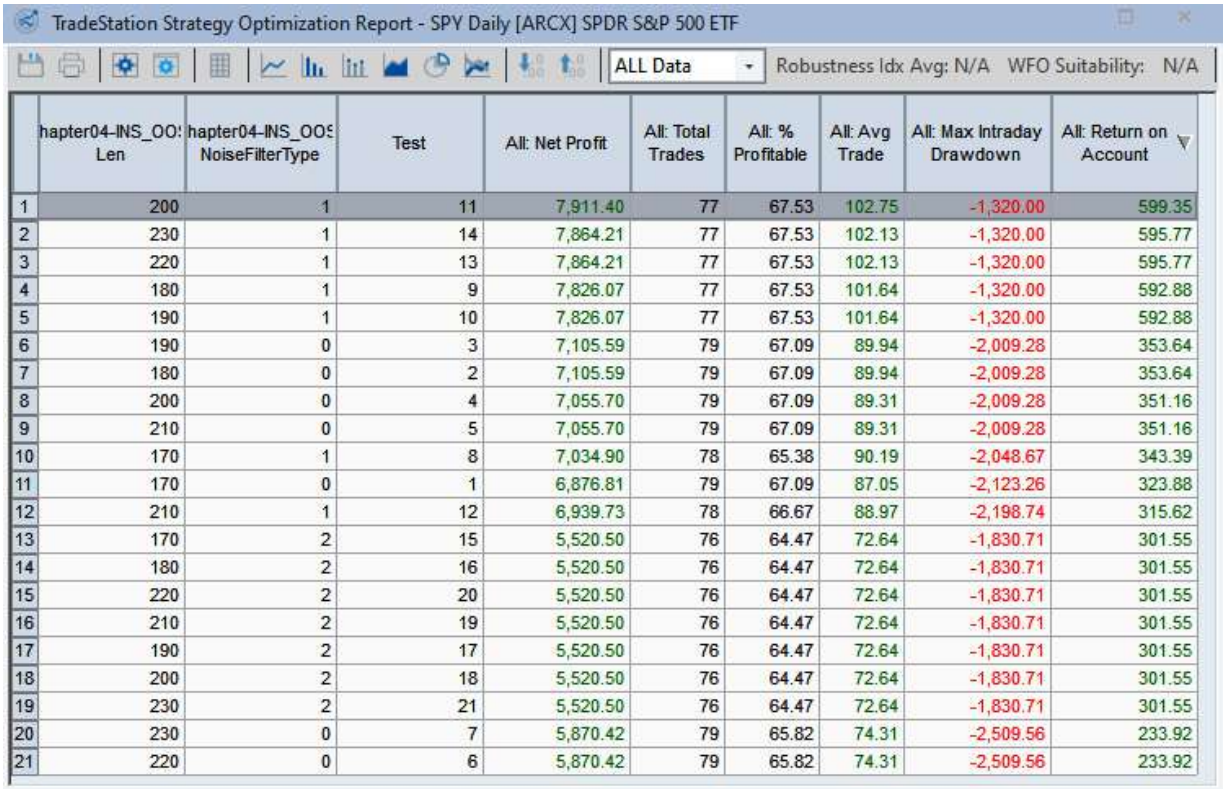
Let’s look at an example of in-sample, out-of-sample validation. Let’s say we want to train the strategy on the period between January 2013 and December 2019 and then test out-of-sample on the period between January 2020 and March 2024.

In order to achieve this task, we need to follow these steps:

1. Set **`StartDate_=1130101`** (January 1, 2013).

2. Set **`EndDate =1191231`** (December 31, 2019).
3. Run an optimization.
4. Select what you think is the most appropriate set of parameters.
5. Re-launch backtesting with the selected set of parameters by changing **`StartDate=1200101`** (January 1, 2020) and **`EndDate_=1240229`** (February 29, 2024).

In *Figure 14.21*, you can see the Optimization Report related to the in-sample period by varying the moving average length from **`170`** to **`230`** in increments of **`10`** and **`NoiseFilterType`** from **`0`** to **`2`** (see *Chapter 3* for reference) for a total of 21 runs.


TradeStation Strategy Optimization Report - SPY Daily [ARCX] SPDR S&P 500 ETF
ALL Data
Robustness Idx Avg: N/A
WFO Suitability: N/A

	hapter04-INS_OO: Len	hapter04-INS_OOS NoiseFilterType	Test	All: Net Profit	All: Total Trades	All: % Profitable	All: Avg Trade	All: Max Intraday Drawdown	All: Return on Account
1	200	1	11	7,911.40	77	67.53	102.75	-1,320.00	599.35
2	230	1	14	7,864.21	77	67.53	102.13	-1,320.00	595.77
3	220	1	13	7,864.21	77	67.53	102.13	-1,320.00	595.77
4	180	1	9	7,826.07	77	67.53	101.64	-1,320.00	592.88
5	190	1	10	7,826.07	77	67.53	101.64	-1,320.00	592.88
6	190	0	3	7,105.59	79	67.09	89.94	-2,009.28	353.64
7	180	0	2	7,105.59	79	67.09	89.94	-2,009.28	353.64
8	200	0	4	7,055.70	79	67.09	89.31	-2,009.28	351.16
9	210	0	5	7,055.70	79	67.09	89.31	-2,009.28	351.16
10	170	1	8	7,034.90	78	65.38	90.19	-2,048.67	343.39
11	170	0	1	6,876.81	79	67.09	87.05	-2,123.26	323.88
12	210	1	12	6,939.73	78	66.67	88.97	-2,198.74	315.62
13	170	2	15	5,520.50	76	64.47	72.64	-1,830.71	301.55
14	180	2	16	5,520.50	76	64.47	72.64	-1,830.71	301.55
15	220	2	20	5,520.50	76	64.47	72.64	-1,830.71	301.55
16	210	2	19	5,520.50	76	64.47	72.64	-1,830.71	301.55
17	190	2	17	5,520.50	76	64.47	72.64	-1,830.71	301.55
18	200	2	18	5,520.50	76	64.47	72.64	-1,830.71	301.55
19	230	2	21	5,520.50	76	64.47	72.64	-1,830.71	301.55
20	230	0	7	5,870.42	79	65.82	74.31	-2,509.56	233.92
21	220	0	6	5,870.42	79	65.82	74.31	-2,509.56	233.92

Figure 4.21 – In-sample Optimization Report

The Optimization Report in *Figure 4.21* shows that among the 21 parameter sets analyzed, the one with the highest ROA has **`Length=200`** and **`NoiseFiterType=1`**.

So, let's save these parameters and add the out-of-sample period by setting **`EndDate_=1240229`** (March 29, 2024). The in-sample, out-of-sample equity line is shown in *Figure 4.22*.

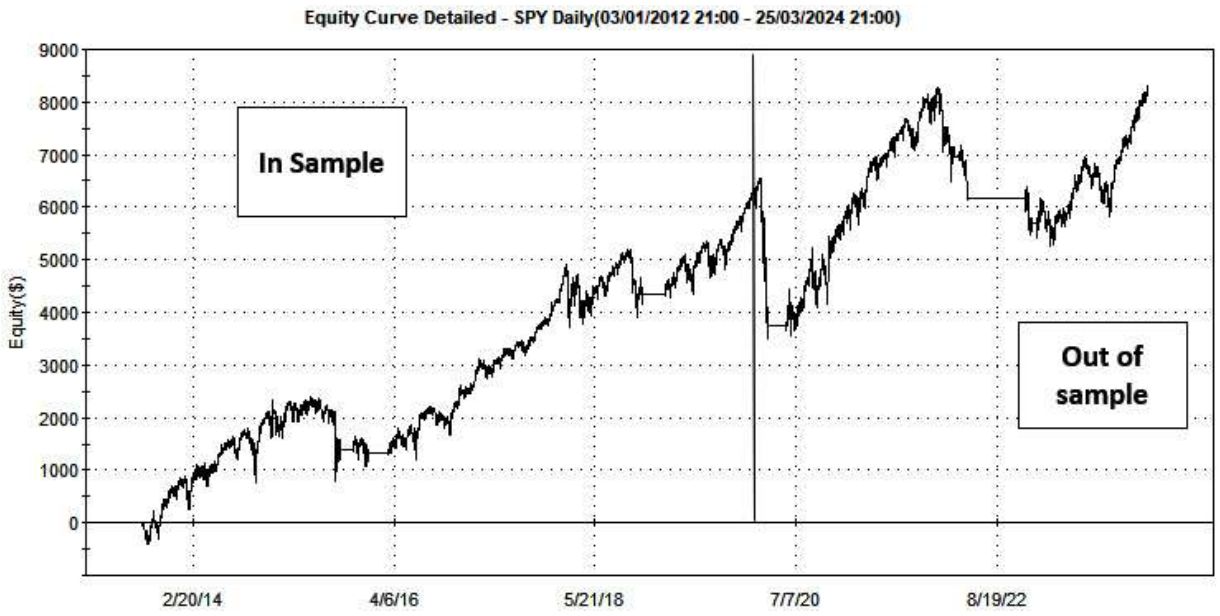


Figure 4.22 – In-sample, out-of-sample equity line report

The in-sample, out-of-sample equity line in *Figure 4.19* shows that, despite two significant drawdowns, the strategy has passed the out-of-sample test and is now very close to a new peak.

Summary

In this chapter, we learned how to evaluate an algorithmic strategy on equities. We emphasized that analyzing financial performance results is not sufficient, as the statistical aspect of algorithmic trading is just as important as the financial one. We highlighted the danger of overfitting and introduced several techniques to mitigate it, such as sensitivity analysis and backtesting on other similar assets. Additionally, we explored how to utilize the EquityEdge Buy&Hold Analyzer as an add-on for quickly evaluating strategies against the underlying asset's buy-and-hold performance. Finally, we discussed the in-sample, out-of-sample process.

It's important to note that the finance industry still lacks a shared and objective method for validating strategies, and research on this matter remains ongoing.

The backtesting and validation strategies are crucial concepts in algorithmic trading, so, re-read and review this chapter if needed. In the next chapter, we will move on to **Forex**.

5

Reversal Strategies

Now that we have gained confidence with backtesting and validating trend strategies, let’s move on to a different trading approach: **reversal**. The goal of this chapter is for you to understand the market rationale behind a reversal strategy, to learn a methodology to select and program technical indicators for such logic, and to program related strategies and backtest them using TradeStation workspaces, charts, the Print Log, and Excel as unique tools for validating multiple assets.

We will cover the following main topics:

- The market rationale behind a reversal strategy
- Writing a reversal strategy
- Backtesting long reversal strategies
- Backtesting short reversal strategies

The market rationale behind a reversal strategy

A **reversal**, also known as a **trend reversal**, happens when the direction of price shifts. These reversals usually unfold gradually over time rather than occurring suddenly. This gradual process occurs because significant investors *require time to enter or exit positions*.

If large investors were to place very large orders on the market all at once, depending on the market liquidity, they could cause significant price movements against them as other market participants would immediately follow their trades.

As a result, large investors must carefully plan and manage the execution of their orders to minimize the impact on the market. This may involve splitting orders into smaller tranches and executing them gradually over time, so as not to move prices too much and obtain a more favorable average price.

This careful management of order execution is crucial for the success of large investors’ operations and requires a deep understanding of market liquidity, as well as advanced skills in financial transaction analysis and execution.

Understanding such behavior, which large investors are forced to undertake because of their dimension, is the key to program reversal algorithms.

There are two types of reversals, both of which we will look at in detail:

- Reversal up
- Reversal down

Reversal up

Reversal up refers to a price movement pattern where a security, commodity, or other asset that has been in a downtrend shows signs of reversing its direction to an upward trend. A reversal up comprises three key steps (*Figure 5.1*):

1. The initial price decline signals a prevailing downtrend in the market.
2. Volatility compression ensues, indicating a phase of consolidation or market uncertainty characterized by decreased volatility and limited price movements.
3. Subsequent price increase suggests that bearish pressure might be dwindling, potentially paving the way for a reversal into an uptrend:

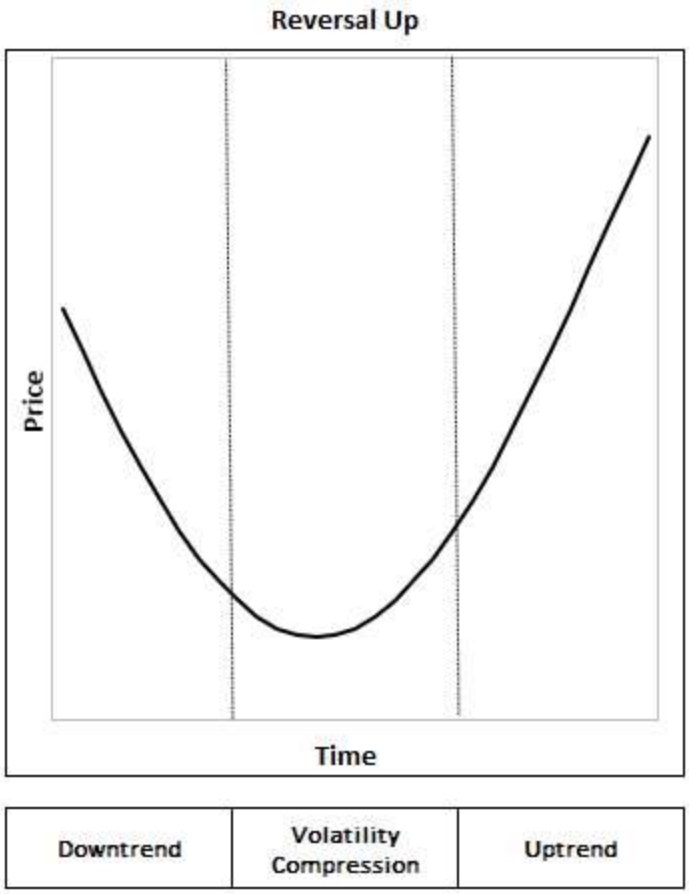


Figure 5.1 – Reversal up

Figure 5.1 illustrates the typical appearance of price charts during reversal up.

Reversal down

Conversely, **reversal down** describes a price movement pattern where a security, commodity, or another asset that has been in an uptrend shows signs of reversing its direction to a downtrend. A

reversal down unfolds in the following three steps (*Figure 5.2*):

1. This is initiated by a price increase, signaling an ongoing uptrend in the market.
2. This is followed by volatility compression, indicating a phase of consolidation or market uncertainty marked by reduced volatility and constrained price movements.
3. This culminates in a price decline, suggestive of diminishing bullish momentum and the potential for the market to reverse direction into a downtrend:

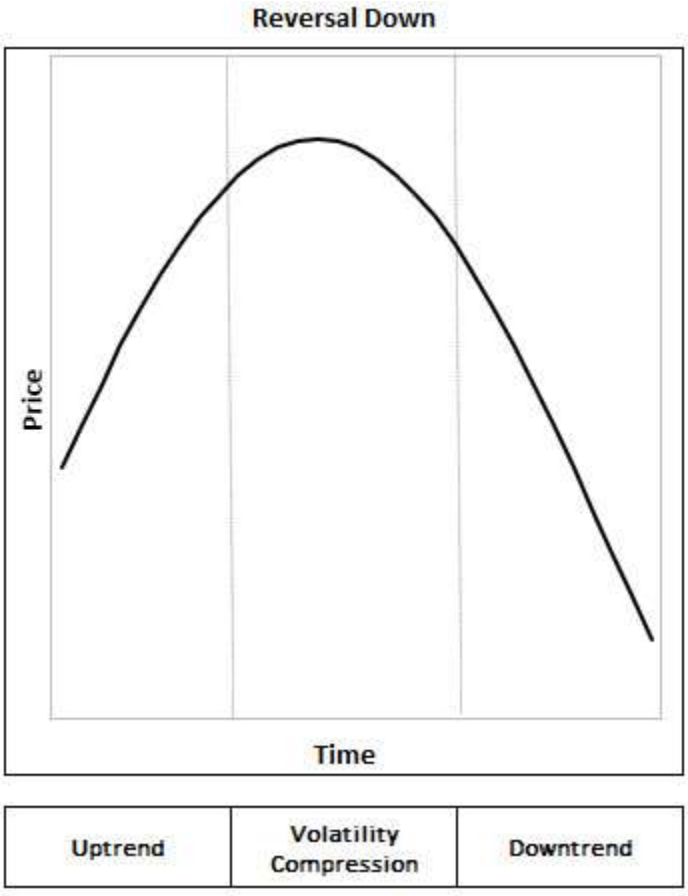


Figure 5.2 – Reversal down

Figure 5.2 illustrates the typical appearance of price charts during reversals. Now, let's move on and learn how to catch such patterns algorithmically.

Writing a reversal strategy

To detect a reversal, we need to write an algorithm for any of the following three components:

- **Existing trend**, where we are going to use the simple moving average
- **Volatility compression**, where we are going to use a *VoltyRatio indicator*
- **Final trend**, where we are going to use a *breakout*

Existing trend

To catch the existing trend, according to what we presented in *Chapter 3*, we know that we can use the simple moving average. So, to catch a downtrend, we can write the following condition:

```
close<average(close,PullLen)
```

Here, **`PullLen`** represents the length to calculate the simple moving average.

Adding such a condition to the long entry strategy's rules will allow long trades only when the price is below the simple moving average selected. Conversely, to detect an existing uptrend, we can write this condition:

```
close>average(close,PullLen)
```

Adding this condition to the short entry strategy's rules will allow short trades only when the price is above the simple moving average selected.

Now, let's learn how to deal with volatility compression.

Volatility compression

Volatility compression can be defined as the period in the financial markets when the range of price movements for a particular asset contracts or narrows significantly.

We can use *VoltyRatio* to detect the volatility compression by writing the following lines of code:

```
variables:
    Volty1( 0 ) ,
    Volty2( 0 ),
    VoltyRatio(0);
Volty1 = Highest(H,Len1)-lowest(L,Len1) ;
Volty2 = Highest(H,Len2)-lowest(L,Len2) ;
if Volty2<>0 then VoltyRatio=Volty1/Volty2;
Plot1( VoltyRatio, "VoltyRatio" ) ;
Plot2( VoltyTrigger, "VoltyTrigger" ) ;
```

Here, **`Volty1`** represents the price range for the shorter time window defined by **`Len1`**, while **`Volty2`** represents the price range for the longer time window defined by **`Len2`**. Note that **`VoltyRatio`** is the ratio between **`Volty1`** and **`Volty2`**.

Figure 5.3 shows the VoltyRatio indicator applied to a chart:

Figure 5.3 – VoltyRatio indicator to detect volatility compression

As you can see, the `VoltyRatio` indicator ranges from 0 to 1. The lower the level, the stronger the volatility compression is. In this chart example, we set 0.3 as a trigger to detect the volatility's compression and color the candles in red any time the indicator goes below 0.3 (ellipses). As you will see in the backtesting sections of this chapter, this trigger can be modified.

Next, we'll consider final directional price move detection using a breakout.

Final trend

To catch the start of a new trend, we can use a breakout. A **breakout** refers to a significant price movement in a particular direction that surpasses a predefined level of *support* or *resistance*:

- **Support** refers to the price level at which a security or market typically encounters buying interest, halting further price declines. It represents the level where demand for the asset is strong enough to overcome selling pressure, causing the price to bounce back up.
- **Resistance**, on the other hand, is the price level at which a security or market typically encounters selling interest, halting further price increases. It represents the level where the supply of the asset is strong enough to overcome buying pressure, causing the price to reverse or stall its upward movement.

In the Technical Analysis library, there are dozens of different ways to detect support and resistance. For this chapter, we are going to choose a logic coherent with the context of the entire script.

Let's learn how to algorithmically detect support and resistance:

```
if VoltyRatio<VoltyTrigger then begin
    Resistance=Highest(H,Len1);
    Support=Lowest(L,Len1);
End;
```

In this example, any time there is a volatility compression, meaning any time `VoltyRatio<VoltyTrigger`, we assume `Resistance` is the highest value of the last `Len1` bars and `Support` is the lowest value of the last `Len1` bars.

So, a breakout on the upside occurs when the price of security closes above a previously established resistance level:

```
if close > Resistance then Buy next bar open;
```

Inversely, a breakout on the downside occurs when the price of security closes below a previously established support level:

```
if close < Support then sellshort next bar open;
```

Now that we have a script to detect a breakout, a question may have arisen in your mind: For how many bars should we wait for the breakout after the volatility compression event occurs? Establishing a timeframe is vital to validate the volatility compression setup. Therefore, it's essential to program a condition that dictates initiating trades within *n* days following the volatility compression event. Let's delve into how to set up this condition effectively:

```
if VoltyRatio<VoltyTrigger then begin
    count=0;
End;
count=count+1;
```

The `count` variable resets to `0` whenever there's a volatility compression and then increments by `1` with each passing bar. Simply put, the `count` variable tracks the number of days that have elapsed since the volatility compression event. So, if we want to trade only within *n* days from the event, we can add the following condition to our condition entry rules, where `WaitBars` is an input:

```
if count<WaitBars
```

The final entry conditions script is as follows:

```
if count<WaitBars then Begin
    if Dir=1 and
        close<average(close,PullLen) and
            close>Resistance
                  then buy qty shares next bar open;
(should be at four spaces more than the previous line)
    if Dir=-1 and
```

```
          close>average(close,PullLen) and
              close<Support
                   then sellshort qty shares next bar open;
  End;
```

In the preceding code, the **`if count<WaitBars`** instruction ensures that a volatility compression event occurred in the last **`WaitBars`**. The **`Dir=1`** condition allows us to test only the long side of the strategy, while the **`Dir=-1`** condition allows us to test only the short side of the strategy. The **`close<average(close,PullLen)`** and **`close>average(close,PullLen)`** instructions ensure that a pullback is in place. The **`close>Resistance`** and **`close<Support`** instructions ensure a breakout. So, all three reversal components are taken into account.

You can find the full strategy code in this book's GitHub repository: https://github.com/PacktPublishing/Tradestation-Easylanguage-for-Algorithmic-Trading.

At this point, I'm sure you are very curious to run some backtesting, aren't you?

Backtesting long strategies

To backtest a long reversal strategy, we can go through a process that's very similar to what we covered in *Chapter 4*.

We will go through the following procedure:

1. Identify stock to start with.
2. Run a multiple-sensitivity analysis on the selected stock.
3. Create a **return on assets** (**ROA**) heatmap in Excel.
4. Select the best input set.
5. Backtest the strategy on the out-of-sample full Dow Jones 30 index.

Instead of focusing solely on the strategy's performance under different market conditions or parameter values, this process will assess how the strategy behaves when applied to different assets. The purpose of this approach is to train the strategy on certain data and then test it on different out-of-sample data.

Identifying stock to start with

To establish a starting point, we'll seek out stock with a moderate beta value. Beta measures the asset's volatility compared to the broader market. Beginning our analysis with stock that has a beta value similar to that of the index enhances the accuracy of the entire validation process. *Figure 5.4* shows how to check the beta in TradeStation RadarScreen:

RadarScreen - Page 1

Timeframe | Studies | Data | Settings

All

	Symbol	Description_	Beta	+
1	Dow Jones 30			
2	BA	Boeing Co	1.53	
3	GS	Goldman Sachs Grp	1.40	
4	DIS	Disney (Walt) Co	1.38	
5	CRM	Salesforce Inc	1.29	
6	DOW	Dow Inc	1.28	
7	AAPL	Apple Inc	1.27	
8	AXP	American Express Co	1.20	
9	CAT	Caterpillar Inc	1.15	
10	AMZN	Amazon.com Inc	1.15	
11	JPM	JPMorgan Chase & Co	1.12	
12	CVX	Chevron Corporation	1.10	
13	INTC	Intel Corp	1.07	
14	NKE	Nike Inc Cl B	1.06	
15	HD	Home Depot Inc	1.01	
16	HON	Honeywell Intl	1.01	
17	MMM	3M Co	0.99	
18	V	Visa Inc	0.95	
19	MSFT	Microsoft Corp	0.88	
20	CSCO	Cisco Systems	0.84	
21	IBM	Intl Business Machines Corp	0.71	
22	MCD	McDonald's Corp	0.71	
23	TRV	The Travelers Companies Inc	0.61	
24	AMGN	Amgen Inc	0.60	
25	KO	Coca-Cola Co	0.58	
26	UNH	UnitedHealth Grp Inc	0.54	
27	JNJ	Johnson & Johnson	0.54	
28	WMT	Walmart Inc	0.49	
29	PG	Procter & Gamble Co	0.43	
30	MRK	Merck & Co	0.41	
31	VZ	Verizon Communications	0.40	
32				

Figure 5.4 – Dow Jones components by beta

In *Figure 5.4*, we can see the Dow Jones stock sorted by beta. For this chapter, we are going to start with HD (Home Depot Inc.) because, with a beta of 1.1, it is right in the middle of the list.

Running a multiple sensitivity analysis on HD (Home Depot Inc.)

Let’s run an optimization with the settings shown in *Table 5.1*:

Inputs	Values
`Dir`	1
`PullLen`	5
`Len1`	From 10 to 20 with an interval of 1
`Len2`	From 30 to 50 with an interval of 1
`VoltyTrigger`	From 0.9 to 1.1 with an interval of 0.1
`WaitBars`	10
`BarsIN`	15

Table 5.1 – Sensitivity analysis settings for HD

The rationale behind this optimization is to discern how altering the three parameters, `Len1`, `Len2`, and `VoltyTrigger`, impacts the performance of the strategy concerning the selected symbol, while keeping the remaining inputs (`PullLen`, `WaitBars`, and `BarsIN`) constant. Essentially, we aim to determine the optimal settings for trading a pullback below the 5-day simple moving average, waiting for 10 bars for a potential breakout, and maintaining positions for 15 days. This step-by-step optimization based on market rationale helps a lot in avoiding overfitting.

Exporting data into Excel and creating an ROA heatmap

By running the optimization according to *Table 5.1*, the TradeStation Print Log will be populated by 693 rows that we have to copy and paste into an empty Excel file, as shown in *Figure 5.5*:

	A	B	C	D	E	F	G	H	I	J	K	L
1	Symb	Dir	PullL	Le	Le	VoltyTrigg	WaitBa	Bars	Totaltrad	NetProf	Maxiddrawdow	ROA
680	HD	1	5	14	35	1.1	10	15	4	- 12,889	- 15,550	-0.83
681	HD	1	5	15	34	1.1	10	15	6	- 14,746	- 17,599	-0.84
682	HD	1	5	15	30	1.1	10	15	6	- 16,585	- 19,246	-0.86
683	HD	1	5	14	28	1.1	10	15	5	- 18,505	- 21,166	-0.87
684	HD	1	5	15	28	1.1	10	15	4	- 18,051	- 20,712	-0.87
685	HD	1	5	14	31	1.1	10	15	4	- 16,010	- 18,408	-0.87
686	HD	1	5	14	33	1.1	10	15	5	- 17,144	- 19,805	-0.87
687	HD	1	5	15	35	1.1	10	15	4	- 17,767	- 20,428	-0.87
688	HD	1	5	15	36	1.1	10	15	4	- 17,767	- 20,428	-0.87
689	HD	1	5	14	29	1.1	10	15	5	- 19,206	- 21,867	-0.88
690	HD	1	5	14	34	1.1	10	15	5	- 20,364	- 23,025	-0.88
691	HD	1	5	15	29	1.1	10	15	4	- 20,554	- 23,215	-0.89
692	HD	1	5	15	37	1.1	10	15	5	- 22,783	- 25,444	-0.9
693	HD	1	5	12	20	1.1	10	15	3	- 8,822	- 8,958	-0.98
694	HD	1	5	15	22	1.1	10	15	1	- 11,151	- 11,151	-1
695												

Figure 5.5 – Excel file with optimization data on HD

Starting from the data in *Figure 5.5*, we can use a PivotTable to reshape data and create an ROA heatmap, as shown in *Figure 5.6*, by following the same procedure that we used in *Chapter 4*:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1																						
2	VoltyTrig	Len2																				
3	Len1	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
4	0.9	1.7	2.9	2.9	2.6	2.5	3.2	4.9	3.4	2.5	2.9	3.2	3.4	3.4	2.8	2.7	2.6	2.4	2.6	2.3	2.4	2.9
5	5	0.3	0.2	0.3	0.8	0.4	0.3	0.8	0.9	1.0	0.9	1.0	2.0	1.0	1.8	1.9	1.9	1.5	1.5	0.9	0.6	0.6
6	6	1.1	1.2	0.9	2.0	1.7	2.5	2.0	1.9	1.5	1.2	1.9	2.1	1.5	1.7	1.7	1.0	0.9	1.2	1.1	1.0	0.8
7	7	0.9	0.1	0.7	0.8	1.1	1.4	0.9	1.0	0.4	1.5	1.3	2.4	1.9	2.0	1.5	1.3	1.6	1.9	2.0	1.6	2.0
8	8	0.2	0.5	1.2	0.9	1.2	1.7	1.3	1.5	0.5	2.1	1.8	1.2	1.4	1.1	0.7	0.7	0.7	3.9	3.9	4.1	6.8
9	9	1.1	3.0	1.2	1.1	2.8	3.0	3.7	1.9	1.9	2.2	1.9	5.6	4.3	0.9	4.8	3.3	3.8	4.9	6.9	6.9	6.9
10	10	3.1	5.0	5.7	2.9	5.0	3.5	3.8	4.5	3.5	3.6	5.0	3.9	3.6	5.3	3.6	1.7	1.7	2.2	2.8	4.3	4.6
11	11	1.9	4.5	5.5	4.9	4.3	6.0	19.3	9.7	8.1	9.0	7.9	6.3	6.6	5.5	4.3	6.5	5.5	5.9	5.0	5.0	3.6
12	12	1.9	5.4	3.9	3.8	4.3	3.1	5.2	8.1	2.4	6.5	4.1	8.1	9.6	5.0	4.3	4.9	2.4	2.9	1.8	2.0	3.1
13	13	2.9	3.0	3.8	2.4	2.8	1.8	5.2	3.1	1.7	1.5	3.3	2.3	3.8	3.0	2.8	2.8	2.4	2.0	0.9	0.9	1.5
14	14	3.1	7.0	6.3	7.6	1.5	7.1	7.1	3.1	1.0	1.7	6.5	2.7	2.6	3.0	2.8	2.2	3.7	1.2	0.4	0.5	2.7
15	15	2.8	2.7	3.1	1.6	3.0	5.5	5.0	2.1	5.2	1.1	1.1	1.4	1.7	1.6	1.7	2.2	2.4	1.1	0.2	0.4	0.4
16	1	0.7	0.3	0.4	0.3	0.7	0.8	0.7	1.1	1.0	1.4	1.0	1.0	0.4	0.3	0.0	0.1	0.5	0.8	0.6	0.6	0.6
17	5	0.6	0.1	0.0	0.3	0.3	0.2	0.6	0.5	0.3	0.6	0.2	0.3	0.3	0.4	0.1	0.1	0.1	0.3	0.2	0.1	0.0
18	6	0.0	0.2	0.4	0.6	0.4	0.3	0.4	0.4	0.5	0.2	0.2	0.1	0.1	0.1	0.3	0.3	0.1	0.1	0.1	0.3	0.0
19	7	0.2	0.6	0.5	0.5	1.0	0.5	0.1	0.3	0.7	1.4	1.4	1.1	0.9	-	0.4	0.5	0.5	0.6	0.5	0.4	0.2
20	8	2.7	2.7	1.8	1.6	1.1	3.4	1.0	1.5	3.6	2.8	2.0	2.9	0.1	0.2	0.0	0.2	0.3	-	0.0	0.6	0.4
21	9	2.9	0.6	0.3	0.6	0.8	0.5	1.5	0.2	0.3	0.7	1.1	1.1	0.2	0.7	0.2	0.1	0.7	1.1	0.5	1.3	1.0
22	10	0.6	0.8	0.7	0.0	1.6	2.3	0.8	2.7	0.5	1.3	0.8	1.2	1.4	0.5	1.0	0.6	1.8	1.7	1.7	0.4	0.7
23	11	0.7	1.1	0.8	0.6	2.3	0.9	0.6	0.3	0.4	1.5	0.8	0.7	1.2	0.6	0.3	0.4	1.0	1.4	1.2	0.5	0.3
24	12	0.3	0.7	0.1	0.0	0.4	0.2	0.1	0.0	0.2	0.5	0.3	1.2	0.3	0.0	0.1	0.0	1.1	2.3	1.8	1.8	1.2
25	13	0.3	0.4	0.2	0.0	0.3	0.2	0.2	0.4	0.5	0.8	0.9	1.0	0.2	0.1	0.1	0.7	0.3	0.8	0.9	0.9	1.0
26	14	0.3	0.3	0.1	0.4	0.6	0.8	0.5	0.9	0.8	1.3	1.3	0.9	0.2	0.1	0.4	0.3	0.5	1.2	1.0	1.2	1.5
27	15	0.1	0.2	0.3	0.9	1.2	1.4	1.8	5.9	4.6	6.0	2.3	1.7	1.5	1.3	0.3	0.3	0.6	1.2	0.8	1.3	1.3
28	1.1	0.1	0.1	0.1	0.5	0.8	0.7	1.2	1.1	0.8	0.9	0.7	0.9	1.1	1.2	0.7	0.6	0.3	0.2	0.4	0.3	0.1
29	5	0.9	0.9	0.5	0.5	1.1	0.1	0.1	0.5	0.1	0.6	0.3	0.3	0.3	0.3	0.8	0.4	0.4	0.4	0.3	0.4	0.5
30	6	0.6	0.2	0.4	0.9	0.2	0.4	1.0	0.5	0.6	0.6	1.0	1.0	0.7	0.5	0.0	0.3	0.8	0.6	0.5	0.4	0.2
31	7	1.0	0.0	0.1	0.2	0.5	3.3	2.9	2.9	2.2	2.0	2.0	0.8	0.8	3.1	0.6	0.0	0.1	0.1	0.3	0.2	0.2
32	8	0.8	0.2	1.8	2.3	3.6	2.5	3.4	1.1	1.3	1.3	0.7	1.0	1.4	1.5	0.8	0.8	0.5	0.6	0.2	0.1	0.1
33	9	0.4	0.2	0.1	0.5	1.8	1.5	4.4	4.3	0.9	0.7	0.6	1.4	1.7	0.8	0.5	0.7	0.2	0.5	0.4	0.1	0.3
34	10	0.8	0.4	0.5	0.0	0.1	0.2	1.2	1.2	1.8	0.4	1.7	4.1	4.6	4.4	4.7	4.4	0.6	0.6	0.6	0.5	0.9
35	11	0.1	0.4	0.5	0.1	1.4	0.4	0.9	0.6	2.3	2.3	2.6	1.4	4.9	4.4	2.1	1.6	1.6	1.6	1.7	0.2	0.2
36	12	1.0	0.9	0.4	1.9	0.8	0.5	0.3	1.0	0.7	2.1	0.8	1.8	0.1	0.1	0.5	0.5	0.5	0.1	0.2	0.1	0.1
37	13	-	0.4	-	0.3	0.6	0.2	0.4	0.1	0.0	1.5	0.4	0.7	0.8	0.8	0.7	0.8	0.3	0.4	0.2	0.4	0.4
38	14	-	-	0.5	0.5	0.5	0.5	0.3	0.1	0.9	0.9	0.3	0.9	0.5	0.9	0.9	0.8	0.6	0.3	2.9	3.5	0.2
39	15	-	-	1.0	-	0.2	-	0.3	0.3	0.9	0.9	0.9	0.4	0.7	0.6	0.8	0.9	0.9	0.9	0.7	0.8	0.3
40																						

Figure 5.6 – ROA heatmap showing multiple-sensitivity analysis results on HD

As you can see, we've configured the conditional formatting to highlight cells where the **ROA** is greater than 4. Such formatting lets us easily see whether and where stable areas of settings are located. This is a measure to increase the strategy's robustness.

Selecting the best parameter set

The task of choosing the best parameters produced by an optimization is very important to avoid overfitting. The trader must ensure that the selected parameters are not the result of chance: for this reason, choosing parameters within a stability range makes the strategy more robust, which means that the backtesting results remain positive even for parameter sets close to the selected one. As observed in *Figure 5.6*, there's a consistent region of stable ROA with a `VoltyTrigger` value of `0.9`, `Len1` ranging between 12 and 20, and `Len2` ranging between 44 and 48.

Let's select the `Len1=14`, `Len2=46`, and `Trigger=0.9` settings as they are in the middle of the positive results.

You can see the equity line of HD in *Figure 5.7*:

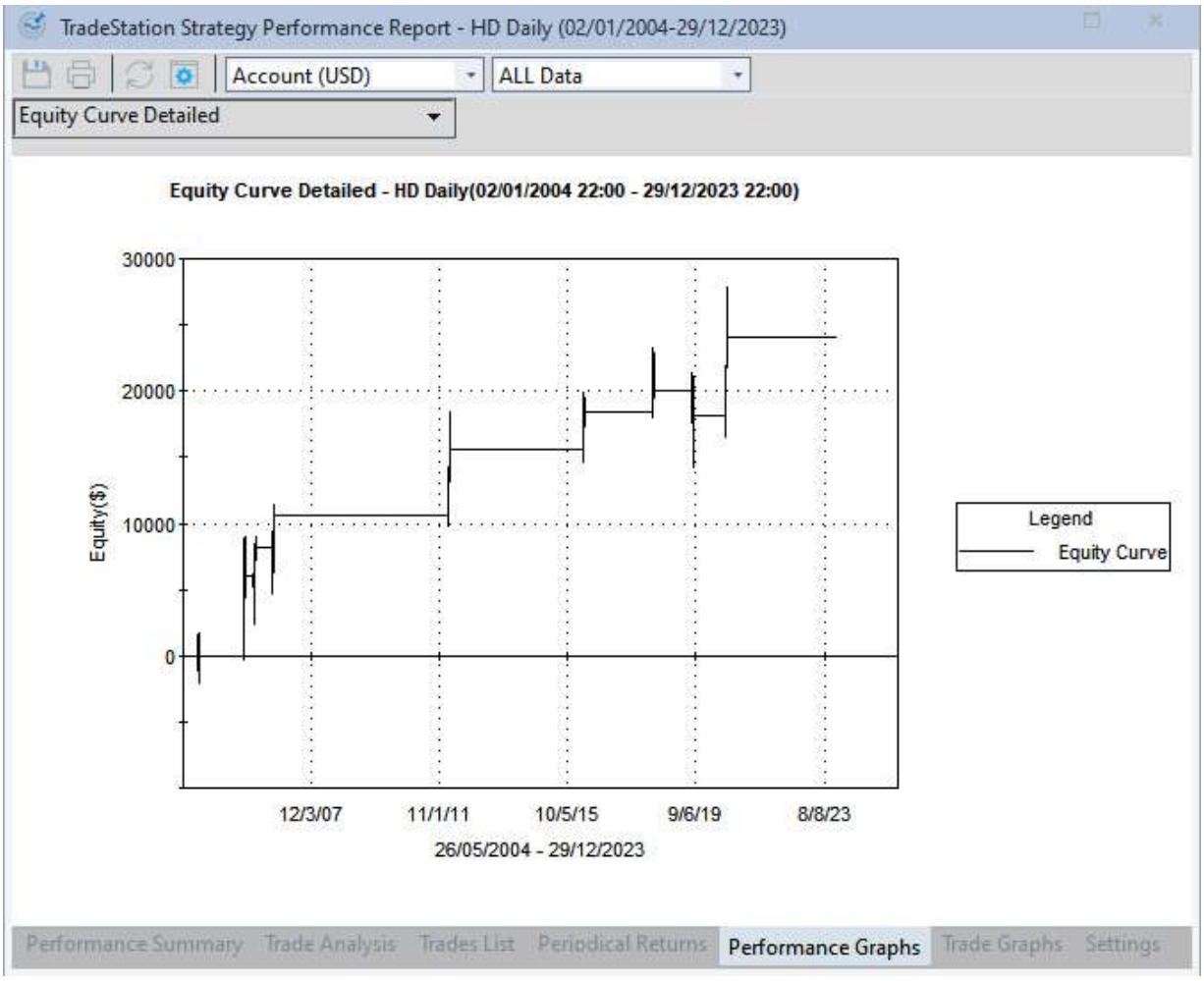


Figure 5.7 – Equity line for HD

Backtesting the strategy on the full Dow Jones 30 index

Keeping the *in-sample*, *out-of-sample* logic we described in *Chapter 4* in mind, we can now run backtesting with the selected parameters on the full list of Dow Jones 30 components. This can give us additional confirmation of the robustness of the strategy. Once backtesting is launched, we can collect the results from the TradeStation Print Log and copy them into Excel (*Figure 5.8*):

	A	B	C	D	E	F	G	H	I	J	K	L
1	symbol	Dir	PullLen	Len1	Len2	VoltyTrigger	WaitBars	BarsIN	Totaltrades	NetProfit	Maxiddrawdown	ROA
2	AMZN	1	5	12	26	0.9	10	15	31	148,539	-12,049	12.33
3	DIS	1	5	12	26	0.9	10	15	15	42,933	-7,350	5.84
4	MSFT	1	5	12	26	0.9	10	15	19	78,748	-14,000	5.63
5	HD	1	5	12	26	0.9	10	15	9	24,039	-5,740	4.19
6	PG	1	5	12	26	0.9	10	15	17	27,600	-6,769	4.08
7	WMT	1	5	12	26	0.9	10	15	13	24,971	-7,032	3.55
8	AXP	1	5	12	26	0.9	10	15	22	51,546	-14,736	3.5
9	NKE	1	5	12	26	0.9	10	15	19	40,560	-15,326	2.65
10	GS	1	5	12	26	0.9	10	15	23	55,576	-21,299	2.61
11	JPM	1	5	12	26	0.9	10	15	19	29,484	-12,531	2.35
12	MCD	1	5	12	26	0.9	10	15	16	15,769	-6,777	2.33
13	AAPL	1	5	12	26	0.9	10	15	26	46,873	-21,334	2.2
14	UNH	1	5	12	26	0.9	10	15	22	19,035	-20,135	0.95
15	VZ	1	5	12	26	0.9	10	15	12	8,379	-9,050	0.93
16	CRM	1	5	12	26	0.9	10	15	26	34,981	-40,018	0.87
17	KO	1	5	12	26	0.9	10	15	15	8,037	-11,361	0.71
18	CVX	1	5	12	26	0.9	10	15	19	13,843	-20,197	0.69
19	BA	1	5	12	26	0.9	10	15	19	17,050	-25,537	0.67
20	DOW	1	5	12	26	0.9	10	15	6	5,938	-9,985	0.59
21	V	1	5	12	26	0.9	10	15	19	7,356	-15,184	0.48
22	HON	1	5	12	26	0.9	10	15	14	5,551	-26,888	0.21
23	INTC	1	5	12	26	0.9	10	15	22	1,022	-29,278	0.03
24	JNJ	1	5	12	26	0.9	10	15	14	-78	-13,562	-0.01
25	AMGN	1	5	12	26	0.9	10	15	11	-1,573	-15,101	-0.1
26	CSCO	1	5	12	26	0.9	10	15	17	-5,888	-17,257	-0.34
27	CAT	1	5	12	26	0.9	10	15	14	-20,913	-47,043	-0.44
28	TRV	1	5	12	26	0.9	10	15	15	-12,683	-25,280	-0.5
29	MRK	1	5	12	26	0.9	10	15	21	-18,385	-28,958	-0.63
30	IBM	1	5	12	26	0.9	10	15	14	-19,361	-28,569	-0.68
31	MMM	1	5	12	26	0.9	10	15	18	-30,733	-34,907	-0.88

Figure 5.8 – Excel file with out-of-sample data on the full Dow Jones list

Figure 5.8 shows the out-of-sample results for the entire Dow Jones list. As you can see, 8 stocks out of 30 show negative results. A question may have arisen in your mind at this point: does the strategy perform the same on stocks with different beta? To answer this question, let's modify the Excel file and add beta cluster analysis (*Figure 5.9*):

	A	I	J	K	L	M	N	O	P	Q	R	S	T
1	symbol	Totaltrades	NetProfit	Maxiddrawdown	ROA	Beta	Beta Cluster						
2	AMZN	31	148,539	-12,049	12.33	1.15	Avg		Beta Cluster				
3	DIS	15	42,933	-7,350	5.84	1.38	High		Low Beta	0	0.7		
4	MSFT	19	78,748	-14,000	5.63	0.88	Avg		Avg Beta	0.7	1.2		
5	HD	9	24,039	-5,740	4.19	1.01	Avg		High Beta	1.2			
6	PG	17	27,600	-6,769	4.08	0.43	Low						
7	WMT	13	24,971	-7,032	3.55	0.49	Low						
8	AXP	22	51,546	-14,736	3.5	1.2	High						
9	NKE	19	40,560	-15,326	2.65	1.06	Avg						
10	GS	23	55,576	-21,299	2.61	1.4	High		Beta Cluster	Avg ROA			
11	JPM	19	29,484	-12,531	2.35	1.12	Avg		Low	100%			
12	MCD	16	15,769	-6,777	2.33	0.71	Avg		Avg	204%			
13	AAPL	26	46,873	-21,334	2.2	1.27	High		High	233%			
14	UNH	22	19,035	-20,135	0.95	0.54	Low		Tot	179%			
15	VZ	12	8,379	-9,050	0.93	0.4	Low						
16	CRM	26	34,981	-40,018	0.87	1.29	High						
17	KO	15	8,037	-11,361	0.71	0.58	Low						
18	CVX	19	13,843	-20,197	0.69	1.1	Avg						
19	BA	19	17,050	-25,537	0.67	1.53	High						
20	DOW	6	5,938	-9,985	0.59	1.28	High						
21	V	19	7,556	-15,184	0.48	0.95	Avg						
22	HON	14	5,551	-26,888	0.21	1.01	Avg						
23	INTC	22	1,022	-29,278	0.03	1.07	Avg						
24	JNJ	14	-78	-13,562	-0.01	0.54	Low						
25	AMGN	11	-1,573	-15,101	-0.1	0.6	Low						
26	CSCO	17	-5,888	-17,257	-0.34	0.84	Avg						
27	CAT	14	-20,913	-47,043	-0.44	1.15	Avg						
28	TRV	15	-12,683	-25,280	-0.5	0.61	Low						
29	MRK	21	-18,385	-28,958	-0.63	0.41	Low						
30	IBM	14	-19,361	-28,569	-0.68	0.71	Avg						
31	MMM	18	-30,733	-34,907	-0.88	0.99	Avg						
32													

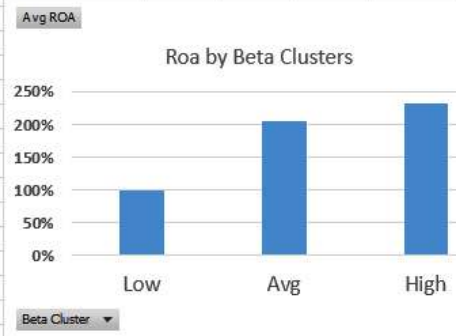


Figure 5.9 – Excel file with out-of-sample results and beta cluster analysis

Figure 5.9 shows the Excel file where we applied the following formulas:

- In cells O3 and R3, we entered the values within which we'll consider beta to be low
- In cells O4 and R4, we entered the values within which we'll consider beta to be average
- Above the value in R4, we'll consider beta to be high
- In column N, we calculated beta clusters
- In cells P10 to O14, we have a PivotTable calculating the average ROA by beta cluster

The result of this analysis is very clear: the higher the beta, the better the out-of-sample performance is.

In this section, we backtested a long reversal strategy on the Dow Jones 30 list. To validate the strategy, we tried a new validation method (in-sample/out-of-sample) by initially training the strategy on a single symbol (we chose HD – Home Depot Inc.) and then testing the optimization results on the entire basket. This approach differs from the one we adopted in *Chapter 4*, where the strategy was built on one historical period and then tested on a different one. Furthermore, we proceeded with a beta cluster analysis on out-of-sample data, which corroborated that the strategy tends to perform more effectively with stocks exhibiting higher volatility.

Backtesting short strategies

Now, we'll attempt to set up a reversal short strategy. Here, we need to make a few considerations: from an algorithmic programming standpoint, a short strategy is simply a long strategy written with conditions reversed. However, from a financial market perspective, it's not the same thing. The stock market trends upward around 90% of the time, so the strategy we're outlining is purely demonstrative.

In this section, we are going to do the following:

- Run a multiple-sensitivity analysis on the **SPDR S&P 500 ETF Trust** (**SPY**)
- Export data into an Excel file and create an ROA heatmap
- Select the best input set

Running multiple-sensitivity analysis on the SPY

Let's run an optimization with the settings shown in *Table 5.2*:

Inputs	**Values**
`Dir`	-1
`PullLen`	5
`Len1`	From 10 to 20 with an interval of 1
`Len2`	From 30 to 50 with an interval of 1
`VoltyTrigger`	From 0.9 to 1.1 with an interval of 0.1
`WaitBars`	10
`BarsIN`	15

Table 5.2 – Sensitivity analysis settings for the SPY

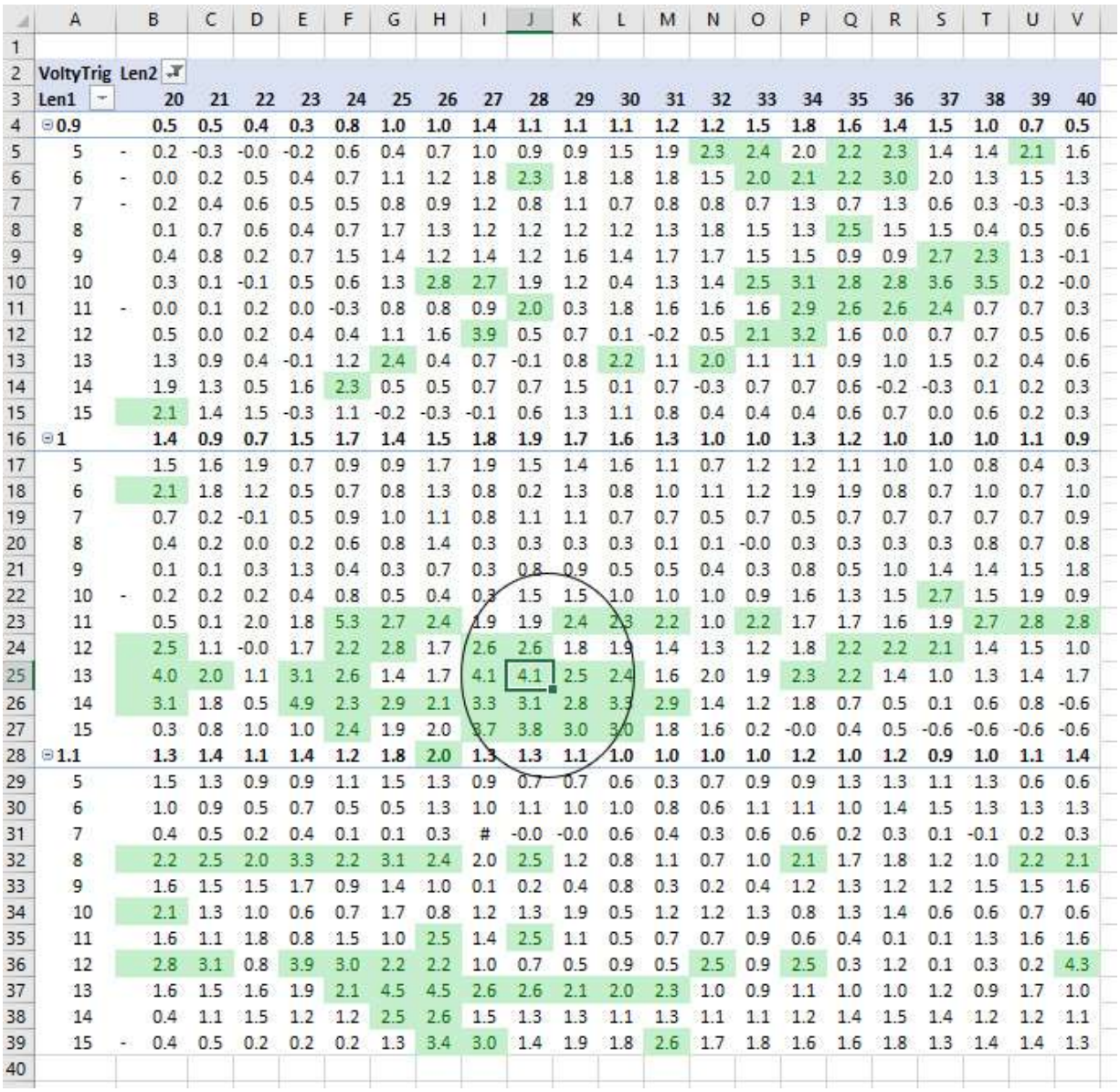
Exporting data into Excel and creating an ROA heatmap

By running the optimization according to *Table 5.2*, the TradeStation Print Log will be populated by 693 rows that we have to copy and paste into an empty Excel file, as shown in *Figure 5.10*.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Symb	Dir	PullL	Le	Le	VoltyTrigg	WaitBa	Bars	Totaltrad	NetProf	Maxiddrawdow	ROA
681	SPY	-1	5	13	39	1.1	10	15	20	22,835	- 13,391	1.71
682	SPY	-1	5	14	39	1.1	10	15	20	20,578	- 16,806	1.22
683	SPY	-1	5	5	40	1.1	10	15	35	12,648	- 22,137	0.57
684	SPY	-1	5	15	39	1.1	10	15	18	22,987	- 16,806	1.37
685	SPY	-1	5	6	40	1.1	10	15	30	20,921	- 15,557	1.34
686	SPY	-1	5	7	40	1.1	10	15	29	7,511	- 23,228	0.32
687	SPY	-1	5	8	40	1.1	10	15	25	23,084	- 11,087	2.08
688	SPY	-1	5	9	40	1.1	10	15	20	18,002	- 11,572	1.56
689	SPY	-1	5	10	40	1.1	10	15	21	8,569	- 14,597	0.59
690	SPY	-1	5	11	40	1.1	10	15	19	15,824	- 10,070	1.57
691	SPY	-1	5	12	40	1.1	10	15	20	35,208	- 8,141	4.32
692	SPY	-1	5	13	40	1.1	10	15	21	18,557	- 18,126	1.02
693	SPY	-1	5	14	40	1.1	10	15	20	19,151	- 17,071	1.12
694	SPY	-1	5	15	40	1.1	10	15	19	21,791	- 17,071	1.28

Figure 5.10 – Excel file with optimization data on the SPY

Starting from the data in *Figure 5.10*, we can use a PivotTable to reshape the data by following the same procedure we followed in *Chapter 4*:



VoltyTrig	Len2																					
Len1		20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
0.9		**0.5**	**0.5**	**0.4**	**0.3**	**0.8**	**1.0**	**1.0**	**1.4**	**1.1**	**1.1**	**1.1**	**1.2**	**1.2**	**1.5**	**1.8**	**1.6**	**1.4**	**1.5**	**1.0**	**0.7**	**0.5**
5	-	0.2	-0.3	-0.0	-0.2	0.6	0.4	0.7	1.0	0.9	0.9	1.5	1.9	2.3	2.4	2.0	2.2	2.3	1.4	1.4	2.1	1.6
6	-	0.0	0.2	0.5	0.4	0.7	1.1	1.2	1.8	2.3	1.8	1.8	1.8	1.5	2.0	2.1	2.2	3.0	2.0	1.3	1.5	1.3
7	-	0.2	0.4	0.6	0.5	0.5	0.8	0.9	1.2	0.8	1.1	0.7	0.8	0.8	0.7	1.3	0.7	1.3	0.6	0.3	-0.3	-0.3
8		0.1	0.7	0.6	0.4	0.7	1.7	1.3	1.2	1.2	1.2	1.2	1.3	1.8	1.5	1.3	2.5	1.5	1.5	0.4	0.5	0.6
9		0.4	0.8	0.2	0.7	1.5	1.4	1.2	1.4	1.2	1.6	1.4	1.7	1.7	1.5	1.5	0.9	0.9	2.7	2.3	1.3	-0.1
10		0.3	0.1	-0.1	0.5	0.6	1.3	2.8	2.7	1.9	1.2	0.4	1.3	1.4	2.5	3.1	2.8	2.8	3.6	3.5	0.2	-0.0
11	-	0.0	0.1	0.2	0.0	-0.3	0.8	0.8	0.9	2.0	0.3	1.8	1.6	1.6	1.6	2.9	2.6	2.6	2.4	0.7	0.7	0.3
12		0.5	0.0	0.2	0.4	0.4	1.1	1.6	3.9	0.5	0.7	0.1	-0.2	0.5	2.1	3.2	1.6	0.0	0.7	0.7	0.5	0.6
13		1.3	0.9	0.4	-0.1	1.2	2.4	0.4	0.7	-0.1	0.8	2.2	1.1	2.0	1.1	1.1	0.9	1.0	1.5	0.2	0.4	0.6
14		1.9	1.3	0.5	1.6	2.3	0.5	0.5	0.7	0.7	1.5	0.1	0.7	-0.3	0.7	0.7	0.6	-0.2	-0.3	0.1	0.2	0.3
15		2.1	1.4	1.5	-0.3	1.1	-0.2	-0.3	-0.1	0.6	1.3	1.1	0.8	0.4	0.4	0.4	0.6	0.7	0.0	0.6	0.2	0.3
1		**1.4**	**0.9**	**0.7**	**1.5**	**1.7**	**1.4**	**1.5**	**1.8**	**1.9**	**1.7**	**1.6**	**1.3**	**1.0**	**1.0**	**1.3**	**1.2**	**1.0**	**1.0**	**1.0**	**1.1**	**0.9**
5		1.5	1.6	1.9	0.7	0.9	0.9	1.7	1.9	1.5	1.4	1.6	1.1	0.7	1.2	1.2	1.1	1.0	1.0	0.8	0.4	0.3
6		2.1	1.8	1.2	0.5	0.7	0.8	1.3	0.8	0.2	1.3	0.8	1.0	1.1	1.2	1.9	1.9	0.8	0.7	1.0	0.7	1.0
7		0.7	0.2	-0.1	0.5	0.9	1.0	1.1	0.8	1.1	1.1	0.7	0.7	0.5	0.7	0.5	0.7	0.7	0.7	0.7	0.7	0.9
8		0.4	0.2	0.0	0.2	0.6	0.8	1.4	0.3	0.3	0.3	0.3	0.1	0.1	-0.0	0.3	0.3	0.3	0.3	0.8	0.7	0.8
9		0.1	0.1	0.3	1.3	0.4	0.3	0.7	0.3	0.8	0.9	0.5	0.5	0.4	0.3	0.8	0.5	1.0	1.4	1.4	1.5	1.8
10	-	0.2	0.2	0.2	0.4	0.8	0.5	0.4	0.8	1.5	1.5	1.0	1.0	1.0	0.9	1.6	1.3	1.5	2.7	1.5	1.9	0.9
11		0.5	0.1	2.0	1.8	5.3	2.7	2.4	1.9	1.9	2.4	2.3	2.2	1.0	2.2	1.7	1.7	1.6	1.9	2.7	2.8	2.8
12		2.5	1.1	-0.0	1.7	2.2	2.8	1.7	2.6	2.6	1.8	1.9	1.4	1.3	1.2	1.8	2.2	2.2	2.1	1.4	1.5	1.0
13		4.0	2.0	1.1	3.1	2.6	1.4	1.7	4.1	4.1	2.5	2.4	1.6	2.0	1.9	2.3	2.2	1.4	1.0	1.3	1.4	1.7
14		3.1	1.8	0.5	4.9	2.3	2.9	2.1	3.3	3.1	2.8	3.3	2.9	1.4	1.2	1.8	0.7	0.5	0.1	0.6	0.8	-0.6
15		0.3	0.8	1.0	1.0	2.4	1.9	2.0	3.7	3.8	3.0	3.0	1.8	1.6	0.2	-0.0	0.4	0.5	-0.6	-0.6	-0.6	-0.6
1.1		**1.3**	**1.4**	**1.1**	**1.4**	**1.2**	**1.8**	**2.0**	**1.3**	**1.3**	**1.1**	**1.0**	**1.0**	**1.0**	**1.0**	**1.2**	**1.0**	**1.2**	**0.9**	**1.0**	**1.1**	**1.4**
5		1.5	1.3	0.9	0.9	1.1	1.5	1.3	0.9	0.7	0.7	0.6	0.3	0.7	0.9	0.9	1.3	1.3	1.1	1.3	0.6	0.6
6		1.0	0.9	0.5	0.7	0.5	0.5	1.3	1.0	1.1	1.0	1.0	0.8	0.6	1.1	1.1	1.0	1.4	1.5	1.3	1.3	1.3
7		0.4	0.5	0.2	0.4	0.1	0.1	0.3	#	-0.0	-0.0	0.6	0.4	0.3	0.6	0.6	0.2	0.3	0.1	-0.1	0.2	0.3
8		2.2	2.5	2.0	3.3	2.2	3.1	2.4	2.0	2.5	1.2	0.8	1.1	0.7	1.0	2.1	1.7	1.8	1.2	1.0	2.2	2.1
9		1.6	1.5	1.5	1.7	0.9	1.4	1.0	0.1	0.2	0.4	0.8	0.3	0.2	0.4	1.2	1.3	1.2	1.2	1.5	1.5	1.6
10		2.1	1.3	1.0	0.6	0.7	1.7	0.8	1.2	1.3	1.9	0.5	1.2	1.2	1.3	0.8	1.3	1.4	0.6	0.6	0.7	0.6
11		1.6	1.1	1.8	0.8	1.5	1.0	2.5	1.4	2.5	1.1	0.5	0.7	0.7	0.9	0.6	0.4	0.1	0.1	1.3	1.6	1.6
12		2.8	3.1	0.8	3.9	3.0	2.2	2.2	1.0	0.7	0.5	0.9	0.5	2.5	0.9	2.5	0.3	1.2	0.1	0.3	0.2	4.3
13		1.6	1.5	1.6	1.9	2.1	4.5	4.5	2.6	2.6	2.1	2.0	2.3	1.0	0.9	1.1	1.0	1.0	1.2	0.9	1.7	1.0
14		0.4	1.1	1.5	1.2	1.2	2.5	2.6	1.5	1.3	1.3	1.1	1.3	1.1	1.1	1.2	1.4	1.5	1.4	1.2	1.2	1.1
15	-	0.4	0.5	0.2	0.2	0.2	1.3	3.4	3.0	1.4	1.9	1.8	2.6	1.7	1.8	1.6	1.6	1.8	1.3	1.4	1.4	1.3

Figure 5.11 – Roa heatmap showing multiple sensitivity analysis results on the SPY

As shown in *Figure 5.11*, we've configured the conditional formatting to highlight cells where the ROA is greater than 3.

Selecting the best parameter set

Observing *Figure 5.11*, there's a consistent region of stable ROA with a **`VoltyTrigger`** value of **`1`**, **`Len1`** ranging between 12 and 15, and **`Len2`** ranging between 27 and 30.

Let's assume the **`Len1=13`**, **`Len2=28`**, and **`Trigger=1`** parameters are the best set because they are in the middle of the positive results. *Figure 5.12* shows the TradeStation equity line:

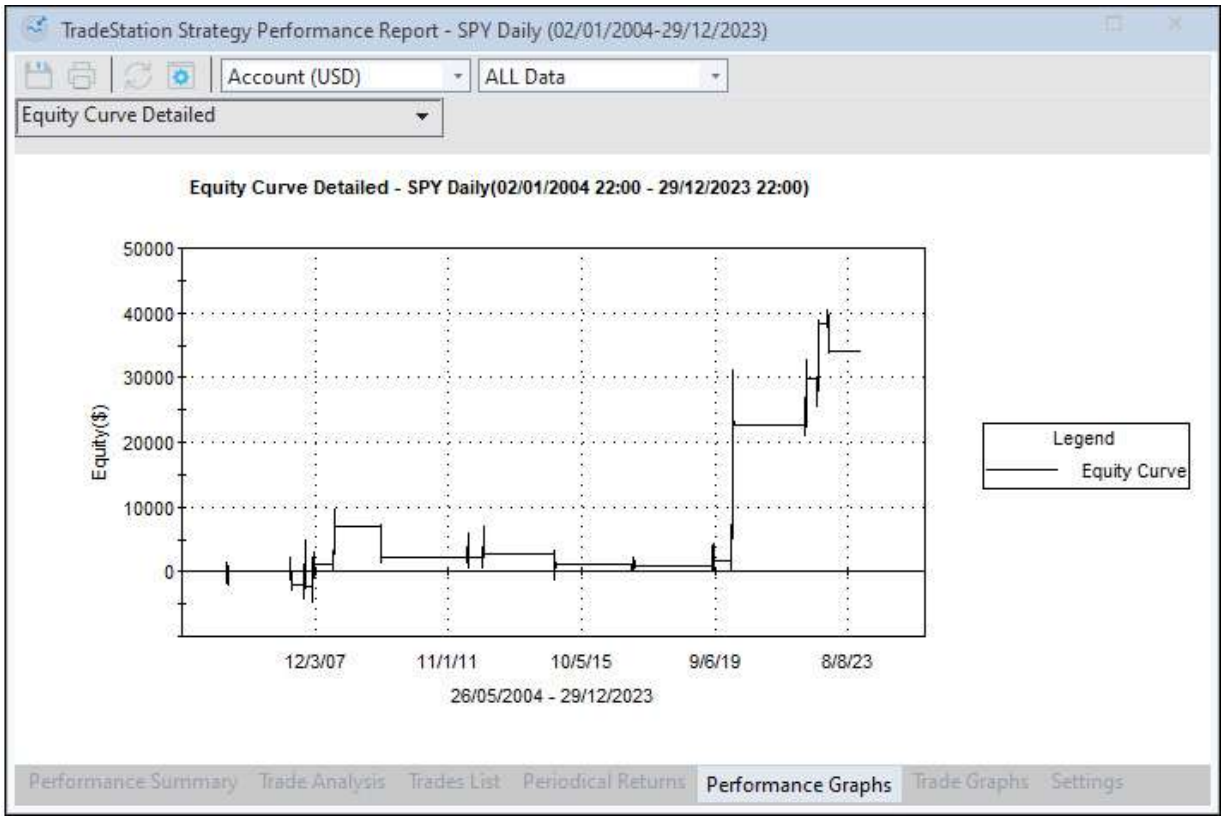


Figure 5.12 – TradeStation equity line for the best parameter set on the SPY

In *Figure 5.12*, you can see the equity line for a reversal short strategy on the SPY. As you can see, the strategy can partially catch the down market of 2008, the crash in 2020 due to COVID, and the down move of 2022.

Whenever we select a set of parameters, it's important to acknowledge that we're inevitably forfeiting potential opportunities presented by the discarded parameter sets. Hence, it might be beneficial to juxtapose equity lines associated with various parameter sets. *Figure 5.13* shows an Excel file displaying the outcomes of optimization, with **Len1** varying between 12 and 14 and **Len2** ranging from 27 to 30, resulting in a total of 16 strategies:

	A	B	C	D	E	F	G
1	Symbol	Len1	Len2	Date	Equity	Strategy	
2	SPY	12	27	1040526	0	12-27	Formula in F2: =+B2&"-"&C2
3	SPY	12	27	1040527	0	12-27	
4	SPY	12	27	1040528	0	12-27	
5	SPY	12	27	1040601	0	12-27	
6	SPY	12	27	1040602	0	12-27	
7	SPY	12	27	1040603	0	12-27	
8	SPY	12	27	1040604	0	12-27	
9	SPY	12	27	1040607	0	12-27	
10	SPY	12	27	1040608	0	12-27	
11	SPY	12	27	1040609	0	12-27	
12	SPY	12	27	1040610	0	12-27	
13	SPY	12	27	1040614	0	12-27	
14	SPY	12	27	1040615	0	12-27	

Figure 5.13 – Excel file with historical performances of 16 short strategies on the SPY

Figure 5.13 shows an Excel file that was obtained by copying and pasting the optimization results. As you can see, we have added column F, with a formula to create the strategy name. We can reshape this data into a PivotTable, as shown in *Figure 5.14*:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1																	
2																	
3	Equity	Strategi															
4	date	12-27	13-27	14-27	12-28	13-28	15-27	14-28	15-28	12-29	13-29	14-29	15-29	12-30	13-30	14-30	15-30
4931	1231220	30,923	34,083	30,089	30,923	34,083	33,097	29,735	32,213	26,656	30,184	29,244	25,503	27,398	30,374	30,248	25,757
4932	1231221	30,923	34,083	30,089	30,923	34,083	33,097	29,735	32,213	26,656	30,184	29,244	25,503	27,398	30,374	30,248	25,757
4933	1231222	30,923	34,083	30,089	30,923	34,083	33,097	29,735	32,213	26,656	30,184	29,244	25,503	27,398	30,374	30,248	25,757
4934	1231226	30,923	34,083	30,089	30,923	34,083	33,097	29,735	32,213	26,656	30,184	29,244	25,503	27,398	30,374	30,248	25,757
4935	1231227	30,923	34,083	30,089	30,923	34,083	33,097	29,735	32,213	26,656	30,184	29,244	25,503	27,398	30,374	30,248	25,757
4936	1231228	30,923	34,083	30,089	30,923	34,083	33,097	29,735	32,213	26,656	30,184	29,244	25,503	27,398	30,374	30,248	25,757
4937	1231229	30,923	34,083	30,089	30,923	34,083	33,097	29,735	32,213	26,656	30,184	29,244	25,503	27,398	30,374	30,248	25,757

Figure 5.14 – 16 equities for short reversal strategies on the SPY

In *Figure 5.14*, you can see the PivotTable reporting the historical equities for the 16 strategies' parameter sets. At this point, we can use a chart to visualize the historical equity line (*Figure 5.15*):

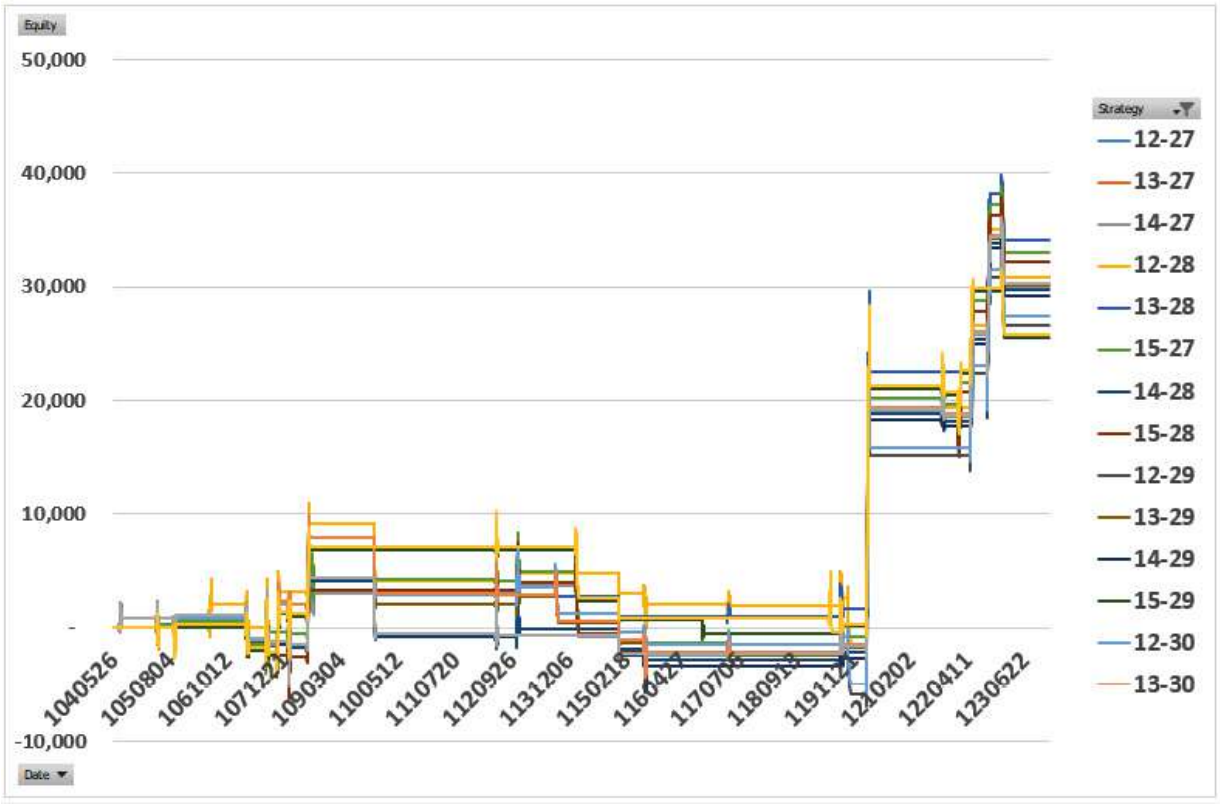


Figure 5.15 – 16 equity lines short reversal strategies on the SPY

In *Figure 5.15*, you can see the 16 equity lines for the short strategies on the SPY. By analyzing this chart, we can see that the 16 lines are very similar to the chart in *Figure 5.10*, confirming the strategy's robustness in the selected inputs' area.

In this section, we delved into employing a reversal short strategy within the stock market. It's crucial to note that short strategies on individual stocks are seldom utilized due to their inherent high risk. Instead, they find their application in stock market indices as portfolio hedging tools. As an example, suppose you possess a $100,000 stock portfolio derived from the S&P 500. It could be judicious to apply short strategies to the benchmark ETF, such as the SPY, either partially (with a size of up to $100,000) or fully (with a size of $100,000). This approach functions akin to insurance against broad market downturns.

Summary

In this chapter, we tackled the theme of reversal, which refers to trend changes. We understood the motivations that lead large operators to consistently adapt the same maneuvers before opening or closing significant financial positions, and that these behaviors can be algorithmically identified. We wrote and tested several long strategies on the Dow Jones 30 basket and some strategies, for educational purposes, on the SPY and learned how to produce and analyze a ROA heatmap in Excel.

In the next chapter, we'll aim to use what we've learned so far to delve deeper into a trend and pullback approach.

6

Trend Pullback Strategies

In the previous chapters, we explained how the financial market moves in **trends** and that every trend eventually ends, leading to a market **reversal**. We then wrote the algorithms for these two aspects of the market separately: trend in *Chapters 3* and *4* and reversal in *Chapter 5*. However, in reality, these two phenomena often occur simultaneously, giving rise to the typical market pattern in waves.

The objective of this chapter is to understand the trend pullback dynamics, how to build some algorithm components suitable for these market phases, and how to use TradeStation and Excel to assemble such components into diversified approaches to trade pullbacks.

In this chapter, we are going to delve into the following:

- The market rationale behind a trend pullback strategy
- Writing trend pullback components
- Assembling trend pullback components
- Conducting sensitivity analysis
- Conducting out-of-sample analysis

> ***NOTE***
>
> *For educational purposes, we will treat both long and short algorithms equally, including both long and short versions in all code. However, in reality, the use of trend pullback strategies is only employed on the long side since the stock market, for structural reasons, tends to grow. For other financial assets, such as commodities and forex, that don't have long bias, as we will see in the upcoming chapters, we will use different techniques.*

The market rationale behind a trend pullback strategy

To understand the logic of trend and pullback, the trader must imagine market movements in terms of waves. Waves in financial markets represent the natural ebb and flow of price movements, influenced by market participants' behaviors. By analyzing these wave patterns, traders and analysts can gain insights into market trends, identify potential reversals, and make more informed trading decisions. Let's have a look at the example in *Figure 6.1*.

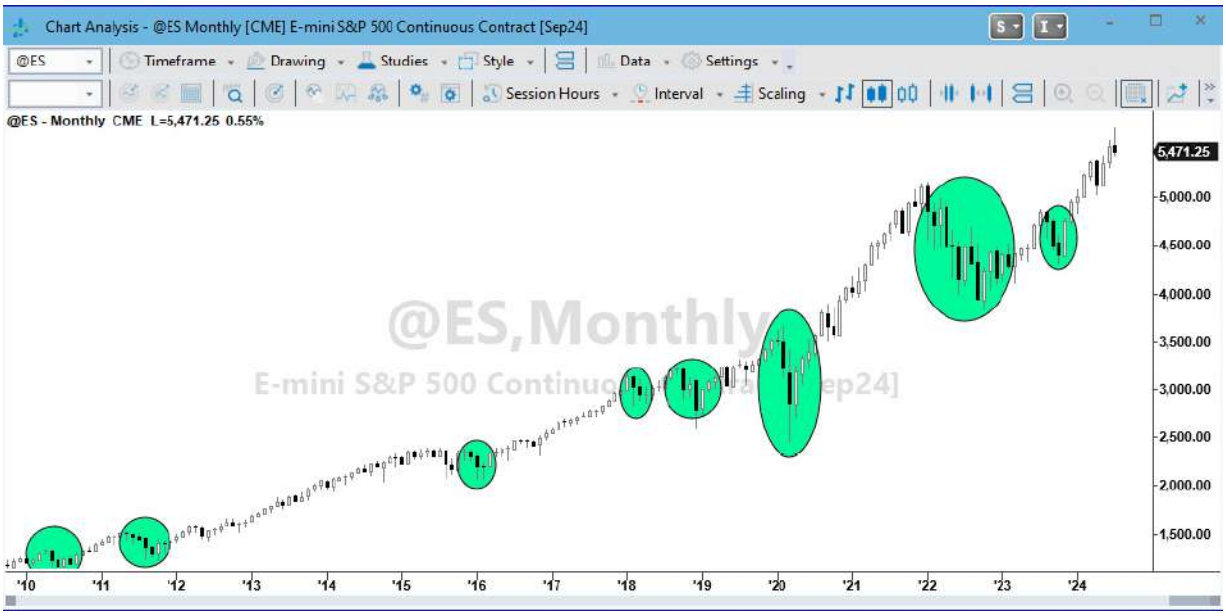


Figure 6.1 – S&P 500 Pullbacks

In *Figure 6.1*, you can see a monthly chart of the main USA stock index, S&P 500, which represents the top 500 stocks listed on USA exchanges. As you can see, despite a strong trend that started in March 2009, the market has formed several waves. These waves are sometimes modest, as shown by the ellipses on the left side of the chart, while other times they are deeper and longer-lasting, like in the ellipses at the right of the chart. Predicting the duration and depth of these movements is challenging, but as you will see, there are regularities in these movements that can be identified and exploited for algorithmic trading. In order to better understand the market behavior, let's learn some definitions.

A **pullback up** is a temporary *decline* in the price of a stock or the broader stock market after a period of upward movement. It's essentially a short-term reversal in the prevailing trend. Pullbacks are often characterized by a decrease in prices over a relatively short period, typically ranging from a few days to a few weeks, before the upward trend resumes. See *Figure 6.2*:

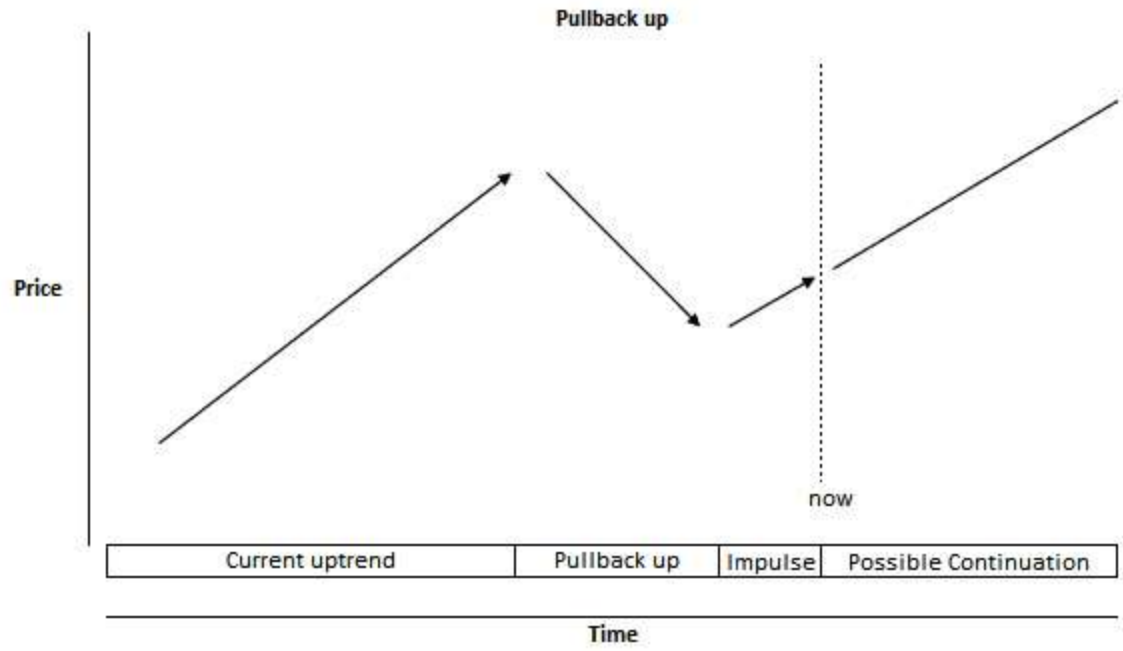


Figure 6.2 – Pullback up

In *Figure 6.2*, you can see a typical price behavior during uptrends.

A **pullback down** is a temporary *upward* movement in the price of a stock or the broader market within the context of an overall downward trend. It's essentially a brief interruption in the downward movement before the prevailing downtrend resumes.

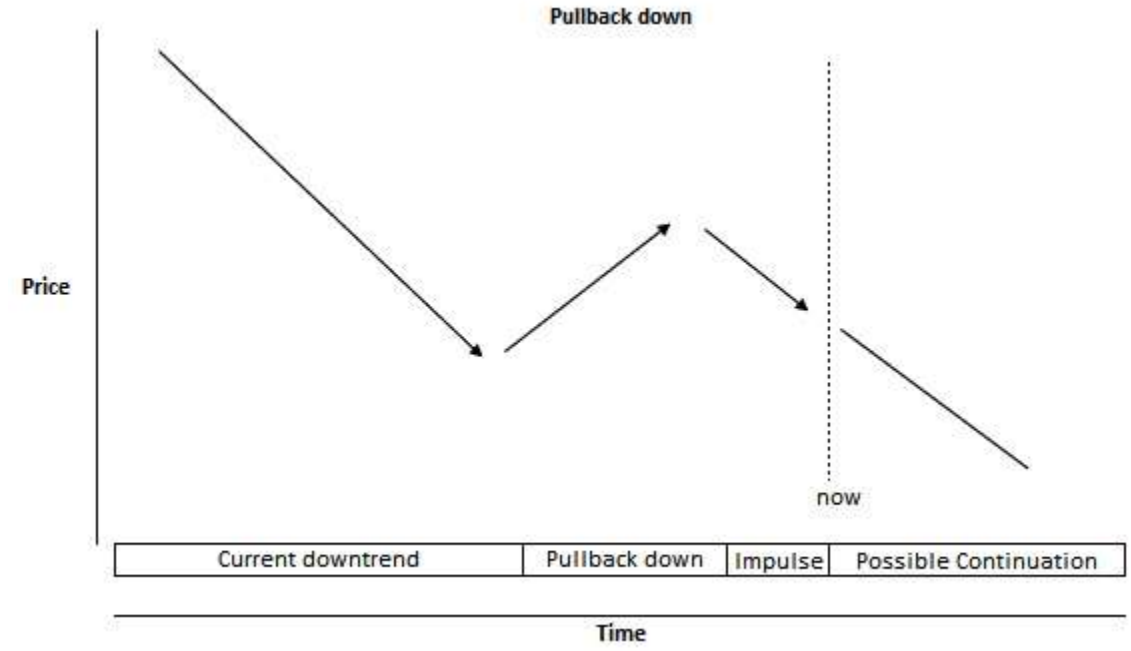


Figure 6.3 – Pullback down

In *Figure 6.3*, you can see a typical price behavior during downtrends.

Pullbacks are indeed a product of the interactions among market participants with diverse time horizons and objectives. Long-term investors often view pullbacks as opportunities to accumulate assets at lower prices, leveraging their focus on fundamental value and their willingness to hold positions over extended periods. In contrast, short-term traders may seize pullbacks as occasions to realize smaller gains or cut losses, reflecting their focus on short-term price movements and technical indicators.

Understanding these distinct motivations and behaviors is crucial for interpreting market trends and effectively navigating through pullbacks. By recognizing how different types of participants respond to market movements, investors can better anticipate price dynamics, adjust their strategies accordingly, and capitalize on opportunities presented by pullbacks. This awareness enables them to manage risk more effectively and align their actions with their investment objectives, whether they prioritize long-term growth, short-term gains, or risk mitigation.

Let's now move on to the development side.

Writing trend pullback components

An algorithm to identify a trend pullback pattern essentially consists of three components:

- An existing trend
- A pullback
- An impulse in the trend direction

Existing trend

In order to catch an existing trend, according to what we presented in *Chapter 3*, we know that we can use simple moving averages. But there are other algorithms, which are a bit more elaborate, to identify a trend, including the following:

- Supertrend
- ADX

Let's delve into these next.

Supertrend

The Supertrend indicator is based on volatility. It consists of two main components: a moving average and a multiplier.

The Supertrend indicator uses a moving average of the asset's price to determine the overall direction of the trend. This moving average can be calculated using different periods, such as 7 days, 14 days, or 50 days, depending on the trader's time horizon.

The function of the multiplier is to adapt the indicator to the volatility of the financial instrument by adjusting the distance of the indicator from the price: the higher the multiplier, the farther the indicator will be from the price.

The resulting Supertrend line is plotted above or below the asset's price chart, depending on whether the trend is bullish or bearish. Traders typically use the crossover of the price and the Supertrend line as potential entry or exit signals. When the price is above the Supertrend line, it may indicate a bullish trend, while a price below the Supertrend line may suggest a bearish trend.

We can write the Supertrend indicator as follows:

```
inputs:Price(close),Supertrend_Len( 22), Supertrend_Multiplier( 20);
variables:    Volty( 0 ),    Supertrend_( 0 );
Volty = Supertrend_Multiplier * AvgTrueRange( Supertrend_Len )[1];
if Price > Supertrend_[1] and Price[1] > Supertrend_[2] then
    Supertrend_ = MaxList( Supertrend_[1], Price - Volty )
else if Price < Supertrend_[1] and Price[1] < Supertrend_[2]
then
    Supertrend_ = MinList( Supertrend_[1], Price + Volty )
else if Price > Supertrend_[1] then
    Supertrend_ = Price - Volty
else
    Supertrend_ = Price + Volty ;
plot1(Supertrend_,"Supertrend",green);if close > Supertrend_ then setplotcolor(1,red);
```

In *Figure 6.4*, you can see the Supertrend indicator applied to a chart.

Figure 6.4 – Supertrend indicator

In *Figure 6.4*, you can see that when the Supertrend indicator is below the price, it is colored green, indicating a bullish trend. Conversely, when the Supertrend indicator is above the price, it is colored red, indicating a bearish trend.

ADX

Another useful indicator to detect a trend is the **Average Directional Index** (**ADX**). It's a technical indicator used to measure the strength of a trend in a financial instrument and was developed by J. Welles Wilder in the late 1970s.

ADX can be interpreted as follows:

- ADX values below a low trigger suggest a weak trend
- ADX values above a high trigger suggest a strong trend

As you can see, ADX provides a trend strength evaluation but tells nothing about direction. To detect trend directions, we need to use some additional information. ADX is generally paired with the **Positive Directional Indicator** (**+DI**) and **Negative Directional Indicator** (**-DI**).

Here's a breakdown of such indicators:

- **`+DI`** measures the strength of the upward movement in price. It's derived from comparing highs.
- **`-DI`** measures the strength of the downward movement in price. It's derived from comparing lows.

In summary, we can say the following:

- A high **`ADX`** with **`+DI`** above **`-DI`** indicates a strong uptrend
- A high **`ADX`** with **`-DI`** above **`+DI`** indicates a strong downtrend
- A low **`ADX`** suggests neither trend is strong

We can write the **`ADX`** indicator as follows:

```
inputs: ADXLength( 14 ),TriggerLevel(40) ;
variables:    double ADXValue( 0 );
ADXValue = ADX( ADXLength );
Plot1( ADXValue,   "ADX" ,red);
if ADXValue>TriggerLevel then setplotcolor(1,blue);
Plot2( TriggerLevel, !( "TrigLevel" ) );
```

Here, **`ADX`** is the TradeStation built-in function, which requires only a length as input. We can write the **`+DI`**/**`-DI`** indicator as follows:

```
inputs:    Len( 14 ) ;
variables:    my_oDMIPlus( 0 ),     my_oDMIMinus( 0 ),     my_oDMI( 0 ),     my_oADX( 0
),     my_oADXR( 0 ),     oVolty( 0 ) ;
Value1 = DirMovement( H, L, C, Len, my_oDMIPlus, my_oDMIMinus, my_oDMI, my_oADX,
my_oADXR, my_oVolty ) ;
plot1(my_odMIPlus,"+DI",green);
plot2(my_odMIMinus,"-DI",red);
```

Here, **`DirMovement`** is the TradeStation built-in function, which requires one length as input and provides both **`odMIPlus`** (**`+DI`**) and **`odMIMinus`** (**`-DI`**).

In *Figure 6.5*, you can see the **`ADX`** indicator applied to a chart:

Figure 6.5 – ADX indicator

In subchart 1 (top) of *Figure 6.5*, the ADX appears as a histogram, with a threshold set at **30**. In subchart 2 (bottom), two lines are visible: the **+DI** line, depicted in green, and the **-DI** line, shown in red. As the ADX surpasses 30, its color changes, indicating the presence of a trend. To determine the direction of this trend, observe the relationship between the **+DI** and **-DI** lines. When the **+DI** line is positioned over the **-DI** line, it signifies a rising trend. Conversely, if the **-DI** line is over the **+DI** line, it indicates a falling trend (as depicted in the rectangle).

Pullbacks

In order to catch pullbacks, we can use several oscillators, including the following:

- RSI
- Stochastic

Let’s delve into these in more detail.

RSI

Relative Strength Index (**RSI**) is a popular technical analysis tool used by traders to measure the magnitude of recent price changes. RSI is calculated by comparing the average gains and losses over a specified time period. The result is a value that ranges from 0 to 100.

TradeStation provides a built-in function to calculate RSI. Here is the indicator's script:

```
inputs:
    Price( Close ),    Length( ) ,    OS_Level( 30 ) ,    OB_Level( 70
    ) ,OS_Color( Cyan ) ,OB_Color( Red )  ;
variables:
    RSIValue( 0 ) ;
RSIValue = RSI( Price, Length ) ;
Plot1( RSIValue, !( "RSI" ) ) ;
Plot2( OB_Level, !( "OB" ) ) ;
Plot3( OS_Level, !( "OS" ) ) ;
if RSIValue > OB_Level then
    SetPlotColor( 1, OB_Color)
else if RSIValue < OS_Level then
    SetPlotColor( 1, OS_Color) ;
```

Here, the **`RSI`** command calls the built-in TradeStation function.

In order to identify a pullback up, we can write the following condition:

```
if RSI(close, RSILen) < oversold
```

Conversely, in order to identify a pullback down, we can write the following:

```
if RSI(close, RSILen) > overbought
```

Here, **`close`** and **`RSILen`** are the two inputs required by the function, and **`oversold`** and **`overbought`** are the triggers to activate the entry conditions.

> ***NOTE***
>
> ***`oversold`*** *refers to a situation where market participants may have driven the asset value to an excessive degree, possibly generating purchase opportunities for traders looking to capitalize on a potential rebound in price.*
>
> ***`overbought`*** *refers to a situation where market participants may have pushed the asset price up to an excessive degree, potentially creating selling opportunities.*

In *Figure 6.6*, you can see an RSI oscillator applied to a chart:

Figure 6.6 – RSI oscillator

In *Figure 6.6*, you can see an RSI oscillator based on a seven-day length, where `oversold` = **30** and `overbought` = **70**. As observed, the indicator changes color based on whether it is above the `overbought` level or below the `oversold` level.

RSI is very useful for catching sharp movements in both directions and the `oversold`/`overbought` triggers help to easily determine how profound the movement should be. For example, an `oversold` level of `20` will require a deeper price-down movement than an `oversold` level of `50`, and so on.

Stochastic

The **Stochastic** oscillator is one of the best algorithms to detect overbought or oversold situations in a trading instrument. It measures the current closing price of a security against its price range over a specific period. The Stochastic oscillator consists of two lines: the **raw line** and a **smoothed line**.

Traders use the Stochastic oscillator to detect potential reversal points in the market. When the oscillator crosses above a certain threshold, it suggests that the asset could experience a downward correction. Conversely, when the oscillator crosses below a lower threshold, it indicates that the asset could experience an upward correction.

We can write the Stochastic oscillator as follows:

```
inputs: PriceH( High ) ,    PriceL( Low ),     PriceC( Close ) ,StochLength( 14 )
,SmoothingLength1( 3 ) ,    SmoothingLength2( 3 ) ,SmoothingType( 1 ) ,    OverSold( 20
```

```
),Overbought(80);
variables:    ReturnValue( 0 ),    oFastK( 0 ),     oFastD( 0 ),     oSlowK( 0
),     oSlowD( 0 ) ;
ReturnValue = Stochastic( PriceH, PriceL, PriceC, StochLength, SmoothingLength1,
SmoothingLength2, SmoothingType, oFastK, oFastD, oSlowK, oSlowD ) ;
Plot1( oSlowK, !( "SlowK" ) ) ;
Plot2( oSlowD, !( "SlowD" ) ) ;
Plot3( OverBought, !( "OverBot" ) ) ;
Plot4( OverSold, !( "OverSld" ) ) ;
```

Here, the **`ReturnValue = Sochastic(…)`** command calls the TradeStation built-in function.

The Stochastic function is a multiple-output function. A **multiple-output** function has two types of parameters: **input** parameters and **input/output** parameters. The values of the input parameters are passed into the multiple-output function, but not modified by the function. The values of the input/output parameters are passed into the multiple-output function and modified by it, and the modified values are then inherited by – or output to – the calling routine.

For example, look at the first lines of the Stochastic function:

```
inputs:
    PriceH         ( numericseries ),
    PriceL         ( numericseries ),
    PriceC         ( numericseries ),
    StochLength    ( numericsimple ),
    Length1        ( numericsimple ),
    Length2        ( numericsimple ),
    SmoothingType  ( numericsimple ),
    oFastK( numericref ),
    oFastD( numericref ),
    oSlowK( numericref ),
    oSlowD( numericref );
```

In this example, the function requires seven inputs (**`numericseries`**) and provides four outputs (**`numericref`**).

In *Figure 6.7*, you can see how the Stochastic oscillator looks when applied to a chart:

Figure 6.7 – Stochastic oscillator

In *Figure 6.7*, you can see a Stochastic oscillator showing sell opportunities when above the `overbought` level (ellipses) and buying opportunities when below the `oversold` level (rectangle).

Final impulse

In order to catch the final impulse toward the existing trend direction, we can use a couple of options:

- A close above/below the previous one
- Oversold/overbought cross

Let's look at these next.

A close above/below the previous one

A basic script to detect an impulse in the long direction is as follows:

```
if close > close[1]
```

Conversely, a script to detect an impulse in the short direction is as follows:

```
if close > close[1]
```

As an alternative to this basic trigger, we can use the same indicator we used to detect the pullback.

Oversold/overbought

For example, to identify an upward impulse following an RSI oversold condition, we can write the following:

```
if RSI(close, RSILen) cross over oversold
```

Here, the `cross over` instruction will require the RSI to have been below the oversold level and then above it in the next bar.

Conversely, to capture a downward impulse following an RSI overbought condition, we can write the following:

```
if RSI(close, RSILen) cross under overbought
```

Here, the `cross under` instruction will require the RSI to have been above the overbought level and then below it in the next bar.

Conversely, if we choose the Stochastic oscillator, we can detect an up impulse by writing the following:

```
if oslowD cross over oversold
```

Conversely, to catch a down impulse, we can write the following:

```
if oslowD cross under overbought
```

Here, `oslowD` is the Stochastic value output.

Assembling components

So far, we have seen that a trend pullback strategy consists of three components: a **trend**, a **pullback**, and a **final movement in the direction of the trend**. We have also seen that each of the three components can be written with different algorithms. It's a bit like having building blocks to assemble together, bearing in mind that any component has its specific purpose.

In *Table 6.1*, you can see nine different combinations regarding the first two main components – trend and pullback:

Algorithms to detect trends	**Algorithms to detect pullbacks**		
	Simple Moving Average (SMA)	**Relative Strength Index (RSI)**	**Stochastic Oscillator**
Simple Moving Average (SMA)	SMA trend + pullback below an SMA	SMA + RSI oversold	SMA + Stochastic oversold

Algorithms to detect trends	**Algorithms to detect pullbacks**		
Supertrend Indicator	Supertrend + pullback below a short SMA	Supertrend + RSI oversold	Supertrend + Stochastic oversold
Average Directional Index (ADX)	ADX + pullback below a short SMA	ADX + RSI oversold	ADX + Stochastic oversold

Table 6.1 – Trend and pullback components

As you can see in *Table 6.1*, given the same rationale, you can write the strategy in different ways depending on the trader's familiarity with technical indicators and, of course, depending on what the back-testing results will tell us.

So, for example, if we identify uptrends with two SMAs, we can write the following three alternatives:

```
if FastSMA > SlowSMA and close < PullBackSMA
if FastSMA > SlowSMA and MyRSI cross over oversold
if FastSMA > SlowSMA and oslowD cross over oversold
```

Here, `FastSMA > SlowSMA` detects the existing uptrend.

If we use the Supertrend indicator, we can write the following three alternatives:

```
if close>Supertrend and close < PullBackSMA
if close>Supertrend and DMIPlus>DMIMinus and MyRSI cross over oversold
if close>Supertrend and DMIPlus>DMIMinus and oslowD cross over oversold
```

Here, `close>Supertrend` detects the existing uptrend.

If we use ADX, we can write the following three alternatives:

```
if ADXValue>AdxTrigger and DMIPlus>DMIMinus and close < PullBackSMA
if ADXValue>AdxTrigger and DMIPlus>DMIMinus and MyRSI cross over oversold
if ADXValue>AdxTrigger and DMIPlus>DMIMinus and oslowD cross over oversold
```

Here, `ADXValue>AdxTrigger and DMIPlus>DMIMinus` detects the existing uptrend.

In the GitHub repository (https://github.com/PacktPublishing/Tradestation-Easylanguage-for-Algorithmic-Trading), you can find the full strategy code.

For the moment, let's select just the following combination:

- Two SMAs to select the trend
- RSI to detect a pullback
- The same RSI oversold cross-over as the entry signal (impulse)

Here is the strategy's script:

```
Input: Dir(1),FastLen(20),SlowLen(200),TradeSize(10000),RSILen(5),RSITrigger(30);
Var: Qty(0);
Qty = TradeSize/Close;
if dir=1 then begin
    If average(close,FastLen) > average(Close,SlowLen)
        and RSI(close,RSILen) cross over RSITrigger
        then Buy ("New Long") Qty Shares next bar open;
End;
if dir=-1 then begin
    If average(close,FastLen) < average(Close,SlowLen)
        and RSI(close,RSILen) cross under RSITrigger
            then Sellshort ("New short") Qty Shares next bar open;
End;
Input:BaRSIN(20);
If BarsSinceentry = BaRSIN then begin
    if mp = 1 then Sell       ("xL") next bar open;
    if mp =-1 then buytocover ("xS") next bar open;
End;
```

In *Figure 6.8*, you can see the entry signal on the **HD** (which stands for Home Depot Inc.) chart:

Figure 6.8 – Trade signals on chart

In *Figure 6.8*, an arrow indicates a new long signal, coinciding with the fast-moving average positioned above the slow-moving average and the RSI crossing the oversold level for the first time. Before proceeding to review performance summaries, it's crucial to ensure the accuracy of entry signals.

In *Figure 6.9*, you can see an equity line result for the applied strategy.

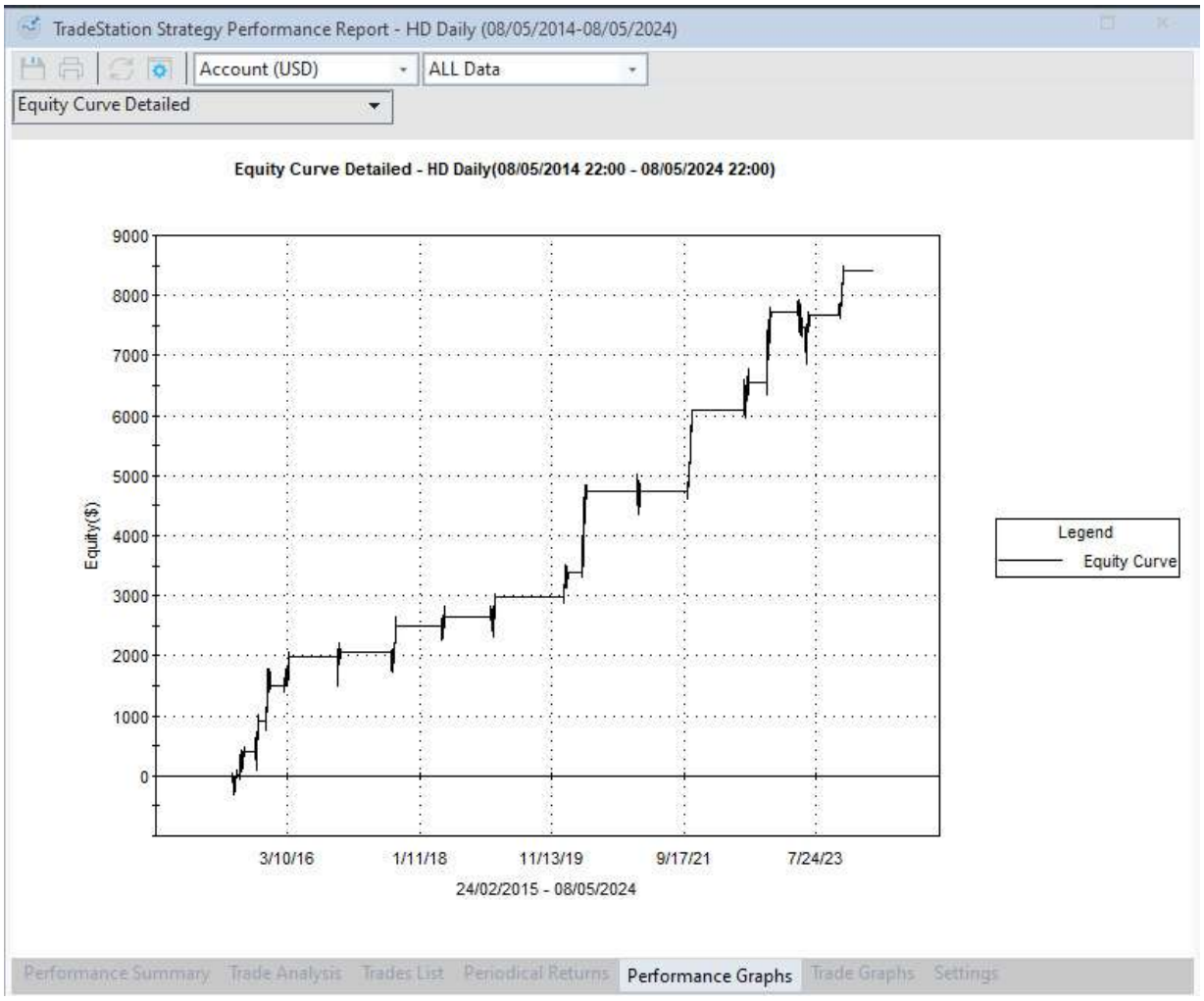


Figure 6.9 – TradeStation equity line

In *Figure 6.9*, you can see the final equity line. Even if the equity looks promising, it means nothing without checking the strategy's robustness. For such purpose, let's now move on to some sensitivity analysis to check robustness of our strategy.

Sensitivity analysis

Compared to the previous chapters, we now have many more parameters in the strategy, so we need to plan how to conduct a sensitivity analysis. You know that the overfitting nightmare is always lurking around the corner, so the way we proceed can make all the difference. For example, we have three different components: trend, pullback, and impulse. Therefore, the best approach is to test them separately. This is why we are going to proceed with two multiple-sensitivity analyses:

- A multiple-sensitivity analysis on the pullback detection using RSI
- A multiple-sensitivity analysis on the trend detection using two SMAs

We don’t need to conduct a sensitivity analysis on the impulse component as it has no setting. So, let’s start.

Running an RSI multiple-sensitivity analysis on HD

To establish a starting point, according to what we described in *Chapter 5*, we are going to start with HD because, if you remember, with a beta of 1.1, it is right in the middle of the Dow Jones 30 list sorted by beta.

Let’s run the first optimization with settings as in *Table 6.2*:

Inputs	**Values**
`Dir`	1
`FastLen`	20
`SlowLen`	200
`RSILen`	From 7 to 50, increment by 1
`RSITrigger`	From 20 to 60, increment by 5
`BaRSIN`	15

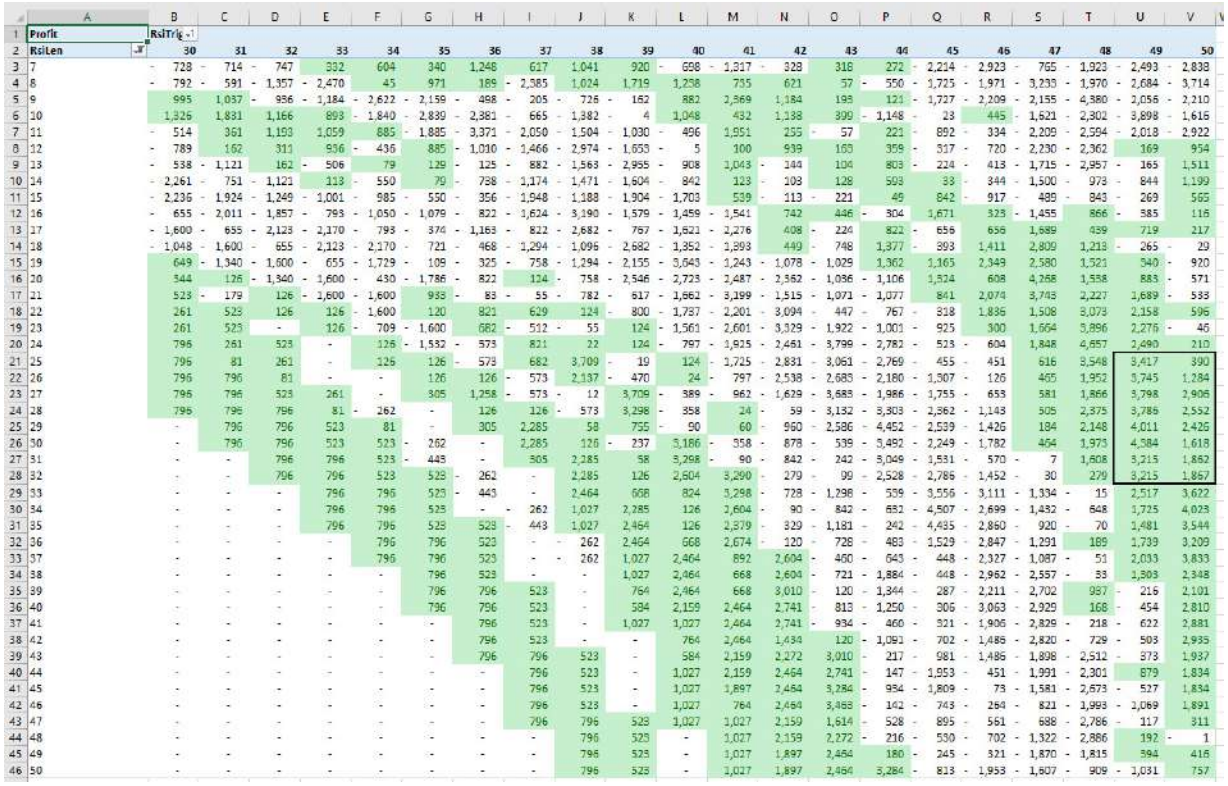
Table 6.2 – RSI sensitivity analysis settings for HD

The rationale behind this optimization, which counts 396 runs, is to discern how altering the two RSI parameters, `RSILen` and `RSITrigger`, impacts the performance of the strategy concerning the selected symbol while keeping the remaining inputs (`FastLen`, `SlowLen`, and `BaRSIN`) constant. Essentially, we aim to determine the optimal settings to identify a pullback in a trend set by the two SMAs (`20` and `200`) and maintain positions for 15 days. This step-by-step optimization based on market rationales helps a lot in avoiding overfitting.

In order to avoid overfitting, we do need sensitivity analysis. By following the same procedure as in *Chapter 5*, we can copy the TradeStation Print Log results and produce the following heat maps:

- Profit heat map to understand how the strategy performs in the inputs range set
- Number of trades heat map to calculate how frequently the strategy trades in the inputs range set
- ROA heat map to select the final best parameters

In *Figure 6.10*, you can see a profit heat map for our benchmark symbol (HD):

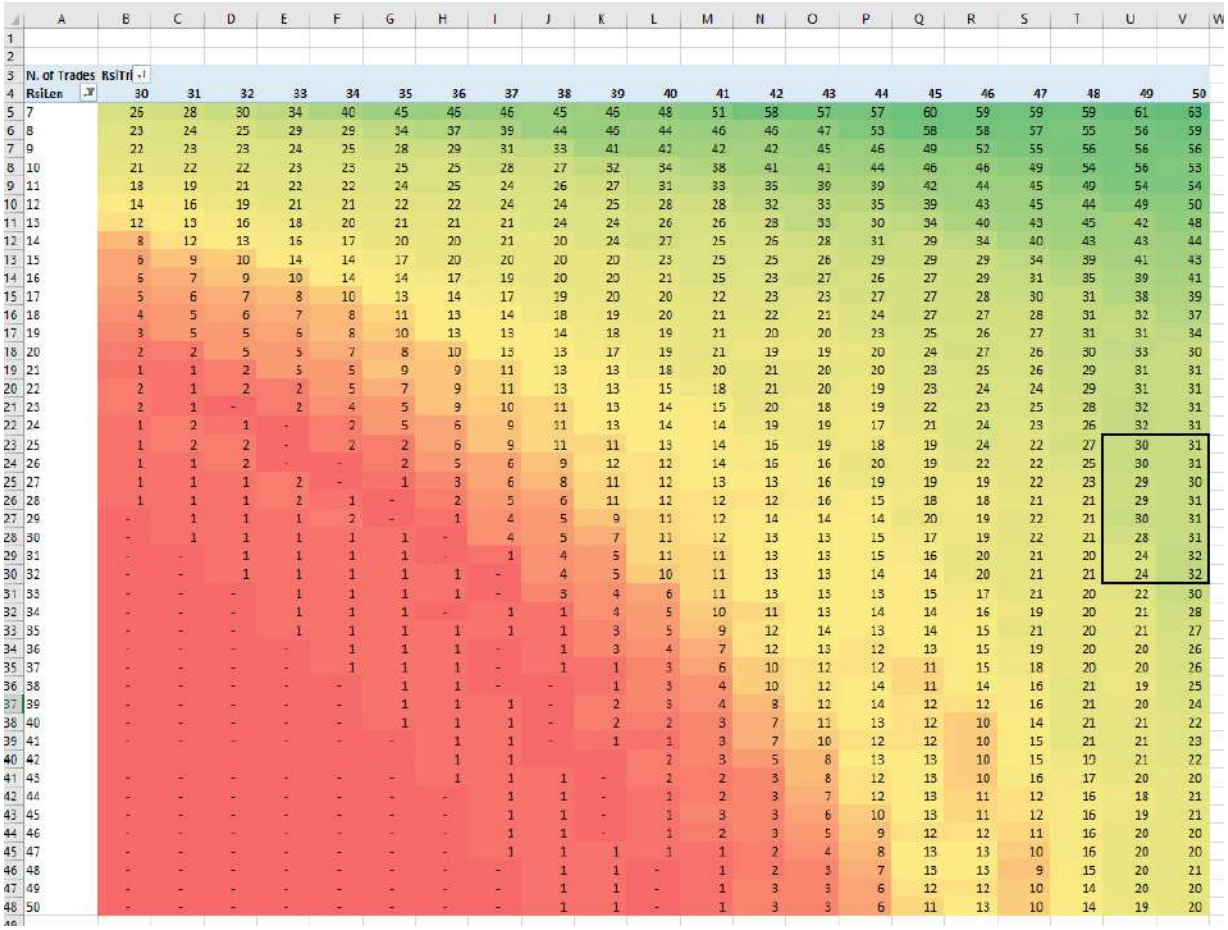


Profit	RsiTrig																					
RsiLen	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
7	-728	-714	-747	332	604	340	1,248	617	1,041	920	-698	-1,317	-328	318	272	-2,214	-2,923	-765	-1,923	-2,493	-2,838
8	-792	-591	-1,357	-2,470	45	971	189	-2,385	1,024	1,719	1,258	735	621	57	-550	-1,725	-1,971	-3,233	-1,970	-2,684	-3,714
9	995	1,037	-936	-1,184	-2,622	-2,159	-498	-205	-726	162	882	2,369	1,184	198	121	-1,727	-2,209	-2,155	-4,380	-2,056	-2,210
10	1,326	1,831	1,166	893	-1,840	-2,830	-2,381	-665	-1,382	-4	1,048	432	1,138	399	-1,148	-28	445	-1,621	-2,302	-3,898	-1,616
11	-514	361	1,193	1,059	885	-1,885	-3,371	-2,050	-1,504	-1,030	-496	1,551	255	-57	221	-892	-334	-2,209	-2,554	-2,018	-2,922
12	-789	162	311	936	-436	885	-1,010	-1,466	-2,974	-1,653	-5	100	939	163	358	-317	-720	-2,230	-2,362	169	954
13	-538	-1,121	162	-506	79	129	-125	-882	-1,563	-2,955	-908	-1,043	-144	104	803	-224	-413	-1,715	-2,957	-165	1,511
14	-2,261	-751	-1,121	113	-550	79	-738	-1,174	-1,471	-1,604	-842	-123	-108	128	593	-33	-344	-1,500	-973	-844	-1,199
15	-2,236	-1,924	-1,249	-1,001	-985	-550	-356	-1,948	-1,188	-1,904	-1,703	-539	-113	-221	-46	-842	-917	-489	-843	-269	-565
16	-655	-2,011	-1,857	-793	-1,050	-1,079	-822	-1,624	-3,190	-1,579	-1,459	-1,541	-742	-446	-304	-1,671	-323	-1,455	-866	-585	-116
17	-1,600	-655	-2,123	-2,170	-793	-374	-1,163	-822	-2,682	-757	-1,621	-2,276	-408	-224	-822	-656	-656	-1,689	-459	-719	-217
18	-1,048	-1,600	-655	-2,123	-2,170	-721	-468	-1,294	-1,096	-2,682	-1,352	-1,893	-449	-748	-1,377	-893	-1,411	-2,809	-1,213	-265	-29
19	649	-1,340	-1,600	-655	-1,729	-109	-325	-758	-1,294	-2,155	-3,643	-1,243	-1,078	-1,029	-1,362	-1,165	-2,349	-2,580	-1,521	-540	-920
20	544	126	-1,340	-1,600	-430	-1,786	-822	124	-758	-2,546	-2,723	-2,487	-2,362	-1,036	-1,106	-1,824	-608	-4,268	-1,558	-883	-571
21	523	-179	126	-1,600	-1,600	933	-83	-55	-782	-617	-1,662	-3,199	-1,515	-1,071	-1,077	-841	-2,074	-3,743	-2,227	-1,689	-533
22	261	523	126	126	-1,600	120	821	629	124	-800	-1,787	-2,201	-3,094	-447	-767	-318	-1,836	-1,508	-3,073	-2,158	-596
23	261	523	-	126	-709	-1,600	682	-512	-55	124	-1,561	-2,601	-3,329	-1,922	-1,001	-925	-300	-1,664	-3,896	-2,276	-46
24	796	261	523	-	126	-1,532	-573	821	22	124	-797	-1,925	-2,461	-3,799	-2,782	-523	-604	-1,848	-4,657	-2,490	-210
25	796	81	261	-	126	126	-573	682	3,709	-19	124	-1,725	-2,831	-3,061	-2,769	-455	-451	-616	-3,548	-3,417	-390
26	796	796	81	-	-	126	126	-573	2,137	-470	24	-797	-2,538	-2,683	-2,180	-1,307	-126	-465	-1,952	-3,745	-1,284
27	796	796	523	261	-	305	1,258	-573	-12	3,709	-389	-962	-1,629	-3,683	-1,986	-1,755	-653	-581	-1,866	-3,798	-2,906
28	796	796	796	81	-262	-	126	126	-573	3,298	-358	-24	-59	-3,132	-3,803	-2,362	-1,143	-505	-2,375	-3,786	-2,552
29	-	796	796	523	81	-	305	2,285	58	796	-90	-60	-960	-2,586	-4,452	-2,539	-1,426	-184	-2,148	-4,011	-2,426
30	-	796	796	523	523	-262	-	2,285	126	-237	-3,186	-358	-878	-539	-3,492	-2,249	-1,782	-464	-1,973	-4,584	-1,618
31	-	-	796	796	523	-443	-	305	2,285	58	-3,298	-90	-842	-242	-3,049	-1,531	-570	-7	-1,608	-3,215	-1,862
32	-	-	796	796	523	523	-262	-	2,285	126	2,604	-3,290	-279	-99	-2,528	-2,786	-1,452	-30	-279	-3,215	-1,857
33	-	-	-	796	796	523	-443	-	2,464	668	824	-3,298	-728	-1,298	-539	-3,556	-3,111	-1,334	-15	-2,517	-3,622
34	-	-	-	796	796	523	-	-262	1,027	2,285	126	-2,604	-90	-842	-682	-4,507	-2,699	-1,432	-648	-3,725	-4,023
35	-	-	-	796	796	523	523	-443	1,027	2,464	126	-2,379	-329	-1,181	-242	-4,435	-2,860	-920	-70	-3,481	-3,544
36	-	-	-	-	796	796	523	-	-262	2,464	668	-2,674	-120	-728	-483	-3,529	-2,847	-1,291	-189	-3,739	-3,209
37	-	-	-	-	796	796	523	-	-262	1,027	2,464	-892	-2,604	-460	-643	-448	-2,327	-1,087	-51	-2,033	-3,833
38	-	-	-	-	-	796	523	-	-	1,027	2,464	668	-2,604	-721	-1,884	-448	-2,962	-2,557	-33	-3,303	-3,348
39	-	-	-	-	-	796	796	523	-	764	2,464	668	-3,010	-120	-1,344	-287	-2,211	-2,702	-987	-216	-2,101
40	-	-	-	-	-	796	796	523	-	584	2,159	2,464	-2,741	-813	-1,250	-306	-3,063	-2,929	-168	-454	-2,810
41	-	-	-	-	-	-	796	523	-	1,027	1,027	2,464	-2,741	-934	-460	-521	-1,906	-2,829	-218	-622	-2,881
42	-	-	-	-	-	-	796	523	-	-	764	2,464	-1,434	-120	-1,091	-702	-1,486	-2,820	-729	-503	-2,935
43	-	-	-	-	-	-	796	796	523	-	584	2,159	-2,272	-3,010	-217	-981	-1,486	-1,898	-2,512	-873	-1,937
44	-	-	-	-	-	-	-	796	523	-	1,027	2,159	-2,464	-2,741	-147	-1,953	-451	-1,991	-2,301	-879	-1,834
45	-	-	-	-	-	-	-	796	523	-	1,027	1,897	-2,464	-3,284	-954	-1,809	-73	-1,581	-2,673	-527	-1,834
46	-	-	-	-	-	-	-	796	523	-	1,027	764	-2,464	-3,468	-142	-743	-264	-821	-1,993	-1,069	-1,891
47	-	-	-	-	-	-	-	796	796	523	1,027	1,027	-2,159	-1,614	-528	-895	-561	-688	-2,786	-317	-311
48	-	-	-	-	-	-	-	-	796	525	-	1,027	-2,159	-2,272	-216	-530	-702	-1,322	-2,886	-192	-1
49	-	-	-	-	-	-	-	-	796	525	-	1,027	-1,897	-2,464	-180	-245	-321	-1,870	-1,815	-594	-416
50	-	-	-	-	-	-	-	-	796	525	-	1,027	-1,897	-2,464	-3,284	-813	-1,953	-1,807	-909	-1,031	-757

Figure 6.10 – Profit heat map – RSI sensitivity analysis

As you can see in *Figure 6.10*, we have configured the conditional formatting to highlight the parameter combinations with a positive profit.

In order to analyze the number of trades, it is very useful to build a number of trades heat map, as shown in *Figure 6.11*.

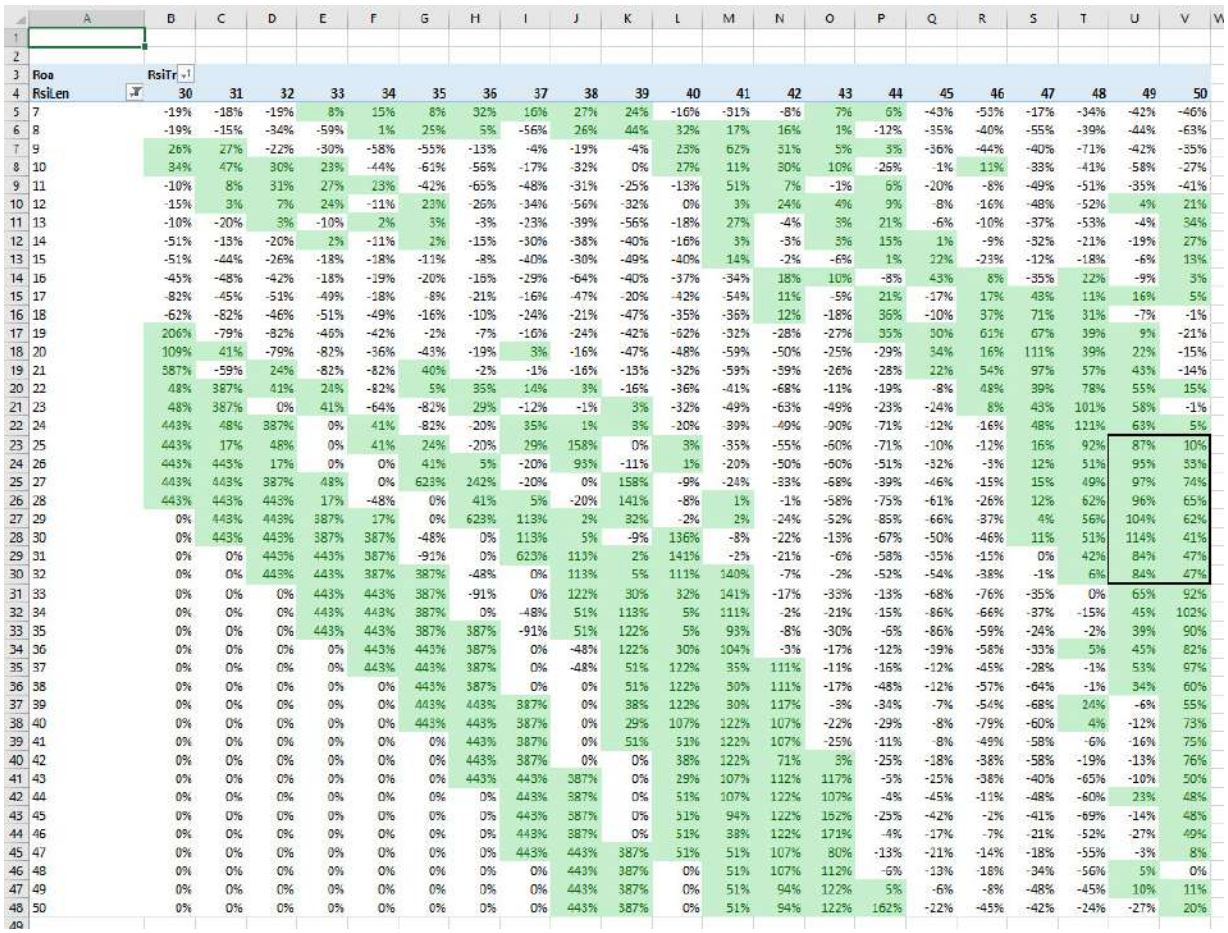


N. of Trades	RsiTr																					
RsiLen	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	
7	26	28	30	34	40	45	46	46	45	46	48	51	58	57	57	60	59	59	59	61	63	
8	23	24	25	29	29	34	37	39	44	46	44	46	46	47	53	58	58	57	55	56	59	
9	22	23	23	24	25	28	29	31	33	41	42	42	42	45	46	49	52	55	56	56	56	
10	21	22	22	23	23	25	25	28	27	32	34	38	41	41	44	46	46	49	54	56	53	
11	18	19	21	22	22	24	25	24	26	27	31	33	35	39	39	42	44	45	49	54	54	
12	14	16	19	21	21	22	22	24	24	25	28	28	32	33	35	39	43	45	44	49	50	
13	12	13	16	18	20	21	21	21	24	24	26	26	28	33	30	34	40	43	45	42	48	
14	8	12	13	16	17	20	20	21	20	24	27	25	25	28	31	29	34	40	43	43	44	
15	6	9	10	14	14	17	20	20	20	20	23	25	25	26	29	29	29	34	39	41	43	
16	5	7	9	10	14	14	17	19	20	20	21	25	23	27	26	27	29	31	35	39	41	
17	5	6	7	8	10	13	14	17	19	20	20	22	23	23	27	27	28	30	31	38	39	
18	4	5	6	7	8	11	13	14	18	19	20	21	22	21	24	27	27	28	31	32	37	
19	3	5	5	6	8	10	13	13	14	18	19	21	20	20	23	25	26	27	31	31	34	
20	2	2	5	5	7	8	10	13	13	17	19	21	19	19	20	24	27	26	30	33	30	
21	1	1	2	5	5	9	9	11	13	13	18	20	21	20	20	23	25	26	29	31	31	
22	2	1	2	2	5	7	9	11	13	13	15	18	21	20	19	23	24	24	29	31	31	
23	2	1	-	2	4	5	9	10	11	13	14	15	20	18	19	22	23	25	28	32	31	
24	1	2	1	-	2	5	6	9	11	13	14	14	19	19	17	21	24	23	26	32	31	
25	1	2	2	-	2	2	6	9	11	11	13	14	16	19	18	19	24	22	27	30	31	
26	1	1	2	-	-	2	5	6	9	12	12	14	16	16	20	19	22	22	25	30	31	
27	1	1	1	2	-	1	3	6	8	11	12	13	13	16	19	19	19	22	23	29	30	
28	1	1	1	2	1	-	2	5	6	11	12	12	12	16	15	18	18	21	21	29	31	
29	-	1	1	1	2	-	1	4	5	9	11	12	14	14	14	20	19	22	21	30	31	
30	-	1	1	1	1	1	-	4	5	7	11	12	13	13	15	17	19	22	21	28	31	
31	-	-	1	1	1	1	-	1	4	5	11	11	13	13	15	16	20	21	20	24	32	
32	-	-	1	1	1	1	1	-	4	5	10	11	13	13	14	14	20	21	21	24	32	
33	-	-	-	1	1	1	1	-	3	4	6	11	13	13	13	15	17	21	20	22	30	
34	-	-	-	1	1	1	-	1	1	4	5	10	11	13	14	14	16	19	20	21	28	
35	-	-	-	1	1	1	1	1	1	3	5	9	12	14	13	14	15	21	20	21	27	
36	-	-	-	-	1	1	1	-	1	3	4	7	12	13	12	13	15	19	20	20	26	
37	-	-	-	-	1	1	1	-	1	1	3	6	10	12	12	11	15	18	20	20	26	
38	-	-	-	-	-	1	1	-	-	1	3	4	10	12	14	11	14	16	21	19	25	
39	-	-	-	-	-	1	1	1	-	2	3	4	8	12	14	12	12	16	21	20	24	
40	-	-	-	-	-	1	1	1	-	2	2	3	7	11	13	12	10	14	21	21	22	
41	-	-	-	-	-	-	1	1	-	1	1	3	7	10	12	12	10	15	21	21	23	
42							1	1			2	3	5	8	13	13	10	15	19	21	22	
43	-	-	-	-	-	-	1	1	1	-	2	2	3	8	12	13	10	16	17	20	20	
44	-	-	-	-	-	-	-	1	1	-	1	2	3	7	12	13	11	12	16	18	21	
45	-	-	-	-	-	-	-	1	1	-	1	3	3	6	10	13	11	12	16	19	21	
46	-	-	-	-	-	-	-	1	1	-	1	2	3	5	9	12	12	11	16	20	20	
47	-	-	-	-	-	-	-	1	1	1	1	1	2	4	8	13	13	10	16	20	20	
48	-	-	-	-	-	-	-	-	1	1	-	1	2	3	7	13	13	9	15	20	21	
49	-	-	-	-	-	-	-	-	1	1	-	1	3	3	6	12	12	10	14	20	20	
50	-	-	-	-	-	-	-	-	1	1	-	1	3	3	6	11	13	10	14	19	20	

Figure 6.11 – Number of trades heat map RSI sensitivity analysis

In *Figure 6.11*, you can see the number of trades heat map, showing that the trades are concentrated in the top-right section of the table, while at the bottom left, there are almost no trades. Such an exercise is useful when you need to analyze a new algorithm behavior; for example, we can understand that the higher the RSI length, the lower the number of trades, and the higher the oversold level, the lower the number of trades.

To see how the parameter settings move the **Return on Assets** (**ROA**), as we have already done in the previous chapters, we can build an ROA heat map, as shown in *Figure 6.12*.



Roa	RsiTr																				
RsiLen	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
7	-19%	-18%	-19%	8%	15%	8%	32%	16%	27%	24%	-16%	-31%	-8%	7%	6%	-43%	-53%	-17%	-34%	-42%	-46%
8	-19%	-15%	-34%	-59%	1%	25%	5%	-56%	26%	44%	32%	17%	16%	1%	-12%	-35%	-40%	-55%	-39%	-44%	-63%
9	26%	27%	-22%	-30%	-58%	-55%	-13%	-4%	-19%	-4%	23%	62%	31%	5%	3%	-36%	-44%	-40%	-71%	-42%	-35%
10	34%	47%	30%	23%	-44%	-61%	-56%	-17%	-32%	0%	27%	11%	30%	10%	-26%	-1%	11%	-33%	-41%	-58%	-27%
11	-10%	8%	31%	27%	23%	-42%	-65%	-48%	-31%	-25%	-13%	51%	7%	-1%	6%	-20%	-8%	-49%	-51%	-35%	-41%
12	-15%	3%	7%	24%	-11%	23%	-26%	-34%	-56%	-32%	0%	3%	24%	4%	9%	-8%	-16%	-48%	-52%	4%	21%
13	-10%	-20%	3%	-10%	2%	3%	-3%	-23%	-39%	-56%	-18%	27%	-4%	3%	21%	-6%	-10%	-37%	-53%	-4%	34%
14	-51%	-13%	-20%	2%	-11%	2%	-15%	-30%	-38%	-40%	-16%	3%	-3%	3%	15%	1%	-9%	-32%	-21%	-19%	27%
15	-51%	-44%	-26%	-18%	-18%	-11%	-8%	-40%	-30%	-49%	-40%	14%	-2%	-6%	1%	22%	-23%	-12%	-18%	-6%	13%
16	-45%	-48%	-42%	-18%	-19%	-20%	-16%	-29%	-64%	-40%	-37%	-34%	18%	10%	-8%	43%	8%	-35%	22%	-9%	3%
17	-82%	-45%	-51%	-49%	-18%	-8%	-21%	-16%	-47%	-20%	-42%	-54%	11%	-5%	21%	-17%	17%	43%	11%	16%	5%
18	-62%	-82%	-46%	-51%	-49%	-16%	-10%	-24%	-21%	-47%	-35%	-36%	12%	-18%	36%	-10%	37%	71%	31%	-7%	-1%
19	206%	-79%	-82%	-46%	-42%	-2%	-7%	-16%	-24%	-42%	-62%	-32%	-28%	-27%	35%	50%	61%	67%	39%	9%	-21%
20	109%	41%	-79%	-82%	-36%	-43%	-19%	3%	-16%	-47%	-48%	-59%	-50%	-25%	-29%	34%	16%	111%	39%	22%	-15%
21	387%	-59%	24%	-82%	-82%	40%	-2%	-1%	-16%	-13%	-32%	-59%	-59%	-26%	-28%	22%	54%	97%	57%	43%	-14%
22	48%	387%	41%	24%	-82%	5%	35%	14%	3%	-16%	-36%	-41%	-68%	-11%	-19%	-8%	48%	39%	78%	55%	15%
23	48%	387%	0%	41%	-64%	-82%	29%	-12%	-1%	3%	-32%	-49%	-63%	-49%	-23%	-24%	8%	43%	101%	58%	-1%
24	443%	48%	387%	0%	41%	-82%	-20%	35%	1%	3%	-20%	-39%	-49%	-90%	-71%	-12%	-16%	48%	121%	63%	5%
25	443%	17%	48%	0%	41%	24%	-20%	29%	158%	0%	3%	-35%	-55%	-60%	-71%	-10%	-12%	16%	92%	87%	10%
26	443%	443%	17%	0%	0%	41%	5%	-20%	93%	-11%	1%	-20%	-50%	-60%	-51%	-32%	-3%	12%	51%	95%	33%
27	443%	443%	387%	48%	0%	623%	242%	-20%	0%	158%	-9%	-24%	-33%	-68%	-39%	-46%	-15%	15%	49%	97%	74%
28	443%	443%	443%	17%	-48%	0%	41%	5%	-20%	141%	-8%	1%	-1%	-58%	-75%	-61%	-26%	12%	62%	96%	65%
29	0%	443%	443%	387%	17%	0%	623%	113%	2%	32%	-2%	2%	-24%	-52%	-85%	-66%	-37%	4%	56%	104%	62%
30	0%	443%	443%	387%	387%	-48%	0%	113%	5%	-9%	136%	-8%	-22%	-13%	-67%	-50%	-46%	11%	51%	114%	41%
31	0%	0%	443%	443%	387%	-91%	0%	623%	113%	2%	141%	-2%	-21%	-6%	-58%	-35%	-15%	0%	42%	84%	47%
32	0%	0%	443%	443%	387%	387%	-48%	0%	113%	5%	111%	140%	-7%	-2%	-52%	-54%	-38%	-1%	6%	84%	47%
33	0%	0%	0%	443%	443%	387%	-91%	0%	122%	30%	32%	141%	-17%	-33%	-13%	-68%	-76%	-35%	0%	65%	92%
34	0%	0%	0%	443%	443%	387%	0%	-48%	51%	113%	5%	111%	-2%	-21%	-15%	-86%	-66%	-37%	-15%	45%	102%
35	0%	0%	0%	443%	443%	387%	387%	-91%	51%	122%	5%	93%	-8%	-30%	-6%	-86%	-59%	-24%	-2%	39%	90%
36	0%	0%	0%	0%	443%	443%	387%	0%	-48%	122%	30%	104%	-3%	-17%	-12%	-39%	-58%	-33%	5%	45%	82%
37	0%	0%	0%	0%	443%	443%	387%	0%	-48%	51%	122%	35%	111%	-11%	-16%	-12%	-45%	-28%	-1%	53%	97%
38	0%	0%	0%	0%	0%	443%	387%	0%	0%	51%	122%	30%	111%	-17%	-48%	-12%	-57%	-64%	-1%	34%	60%
39	0%	0%	0%	0%	0%	443%	443%	387%	0%	38%	122%	30%	117%	-3%	-34%	-7%	-54%	-68%	24%	-6%	55%
40	0%	0%	0%	0%	0%	443%	443%	387%	0%	29%	107%	122%	107%	-22%	-29%	-8%	-79%	-60%	4%	-12%	73%
41	0%	0%	0%	0%	0%	0%	443%	387%	0%	51%	51%	122%	107%	-25%	-11%	-8%	-49%	-58%	-6%	-16%	75%
42	0%	0%	0%	0%	0%	0%	443%	387%	0%	0%	38%	122%	71%	3%	-25%	-18%	-38%	-58%	-19%	-13%	76%
43	0%	0%	0%	0%	0%	0%	443%	443%	387%	0%	29%	107%	112%	117%	-5%	-25%	-38%	-40%	-65%	-10%	50%
44	0%	0%	0%	0%	0%	0%	0%	443%	387%	0%	51%	107%	122%	107%	-4%	-45%	-11%	-48%	-60%	23%	48%
45	0%	0%	0%	0%	0%	0%	0%	443%	387%	0%	51%	94%	122%	162%	-25%	-42%	-2%	-41%	-69%	-14%	48%
46	0%	0%	0%	0%	0%	0%	0%	443%	387%	0%	51%	38%	122%	171%	-4%	-17%	-7%	-21%	-52%	-27%	49%
47	0%	0%	0%	0%	0%	0%	0%	443%	443%	387%	51%	51%	107%	80%	-13%	-21%	-14%	-18%	-55%	-3%	8%
48	0%	0%	0%	0%	0%	0%	0%	0%	443%	387%	0%	51%	107%	112%	-6%	-13%	-18%	-34%	-56%	5%	0%
49	0%	0%	0%	0%	0%	0%	0%	0%	443%	387%	0%	51%	94%	122%	5%	-6%	-8%	-48%	-45%	10%	11%
50	0%	0%	0%	0%	0%	0%	0%	0%	443%	387%	0%	51%	94%	122%	162%	-22%	-45%	-42%	-24%	-27%	20%

Figure 6.12 – ROA heat map showing multiple-sensitivity analysis on RSI

As you can see in *Figure 6.12*, we've configured the conditional formatting to highlight cells where the ROA is greater than 4.

By jointly analyzing *Figure 6.11* (number of trades heat map) and *Figure 6.12* (ROA heat map), we can identify a consistent region of stable ROA, characterized by at least 30 trades, when the RSI length falls within the range of 26 to 32 and is oversold in the range of 49 to 50.

Comparing these two heat maps is invaluable because it can reveal whether you're inadvertently focusing on a strategy with a limited number of trades, which may lack statistical significance. However, it's crucial to strike a balance between the number of trades and strategy performance to avoid overtrading, keeping in mind that even a strategy with a few trades on a single stock may be applicable across the entire stock universe.

Let's select **`Len = 30`** and **`oversold = 40`** as this combination is in the middle of positive results.

Running an SMA multiple-sensitivity analysis on HD

Once we have obtained the best parameter set for RSI, we need to check what's going on with SMA lengths. So, maintaining the best parameter set for RSI, let's run a second optimization with settings as in *Table 6.3*.

Inputs	**Values**
`Dir`	1
`FastLen`	From 3 to 30, increment by 1
`SlowLen`	From 30 to 200, increment by 5
`RSILen`	30
`RSITrigger`	40
`BaRSIN`	15

Table 6.3 – SMA sensitivity analysis settings

By running an optimization with the settings in *Table 6.3*, we can obtain an ROA sensitivity analysis as in *Figure 6.13*.

	A	B	C	D	E	F	G	H	I
1	ROA	FastLen							
2	Slow Len	3	4	5	6	7	8	9	10
3	30	1606%	874%	1952%	597%	532%	546%	546%	559%
4	35	1576%	994%	990%	515%	580%	580%	580%	592%
5	40	853%	967%	756%	524%	561%	530%	530%	101%
6	45	117%	756%	756%	408%	408%	468%	82%	68%
7	50	163%	677%	529%	330%	252%	24%	14%	3%
8	55	72%	504%	648%	57%	57%	27%	3%	38%
9	60	68%	71%	61%	70%	23%	38%	38%	38%
10	65	45%	23%	23%	15%	34%	34%	34%	34%
11	70	0%	28%	-1%	26%	26%	17%	12%	23%
12	75	6%	17%	31%	11%	11%	11%	11%	17%
13	80	22%	12%	6%	6%	-8%	-8%	2%	2%
14	85	7%	17%	11%	22%	13%	12%	13%	2%
15	90	7%	17%	22%	22%	8%	12%	12%	12%
16	95	24%	46%	215%	40%	26%	26%	30%	36%
17	100	29%	46%	215%	40%	69%	55%	55%	55%

Figure 6.13 – ROA sensitivity analysis for SMA

From *Figure 6.13*, we can observe that there is a stable area of ROA with **SlowLen** ranging from 30 to 50 and **FastLen** ranging from 3 to 7.

Let's select the pair **`SlowLen=40`** and **`FastLen=5`**, which is right in the middle of the top-left highlighted area. We now have a final parameter set; so, the next step is to apply it to the entire out-of-sample Dow Jones 30 list.

In the current validation process, we first run a double-sensitivity analysis on RSI parameters while leaving the other strategy settings unaltered. Then, we conduct a double-sensitivity analysis on SMA settings while not altering the other settings. When dealing with multiple-sensitivity analyses, it is advisable to proceed step by step by analyzing pairs of parameters. This two-dimensional analysis allows for easier comprehension of optimization results using heat maps.

Out-of-sample analysis

When we conduct sensitivity analysis, regardless of its complexity, we use in-sample data. On the other hand, we know that to evaluate the robustness of a strategy, it's advisable to conduct an out-of-sample analysis. The purpose of such an analysis is to understand how algorithms perform on unseen data.

In this section, we are going to utilize the parameters obtained from the optimization conducted on a single asset (HD) across the entire Dow Jones 30 index.

In *Figure 6.14*, you can see the results of an out-of-sample analysis on the Dow Jones 30 list.

	symbol	Totaltrades	NetProfit	Maxiddrawdown	ROA	Beta	Beta Cluster
2	AAPL	17	1,485	-1,789	0.83	1.27	High
3	AMGN	20	4,197	-943	4.45	0.6	Low
4	AMZN	12	3,897	-894	4.36	1.15	Avg
5	AXP	22	3,954	-2,286	1.73	1.2	High
6	BA	25	943	-5,238	0.18	1.53	High
7	CAT	24	4,820	-3,570	1.35	1.15	Avg
8	CRM	23	113	-5,659	0.02	1.29	High
9	CSCO	15	2,801	-1,334	2.1	0.84	Avg
10	CVX	24	782	-3,725	0.21	1.1	Avg
11	DIS	22	-685	-4,028	-0.17	1.38	High
12	DOW	14	2,275	-1,698	1.34	1.28	High
13	GS	25	-686	-2,858	-0.24	1.4	High
14	HD	19	6,868	-908	7.56	1.01	Avg
15	HON	27	-403	-3,101	-0.13	1.01	Avg
16	IBM	25	4,303	-1,771	2.43	0.71	Avg
17	INTC	26	4,767	-2,281	2.09	1.07	Avg
18	JNJ	18	102	-1,462	0.07	0.54	Low
19	JPM	20	-919	-3,062	-0.3	1.12	Avg
20	KO	24	368	-2,163	0.17	0.58	Low
21	MCD	19	1,022	-1,439	0.71	0.71	Avg
22	MMM	27	1,370	-3,262	0.42	0.99	Avg
23	MRK	26	727	-2,424	0.3	0.41	Low
24	MSFT	14	4,445	-1,560	2.85	0.88	Avg
25	NKE	23	6,091	-1,360	4.48	1.06	Avg
26	PG	23	3,268	-1,097	2.98	0.43	Low
27	TRV	20	2,606	-2,327	1.12	0.61	Low
28	UNH	15	-133	-3,318	-0.04	0.54	Low
29	V	18	3,887	-1,445	2.69	0.95	Avg
30	VZ	21	-2,804	-4,314	-0.65	0.4	Low
31	WMT	18	3,834	-1,009	3.8	0.49	Low
32							

Beta Cluster		
Low Beta	0	0.7
Avg Beta	0.7	1.2
High Beta	1.2	

Beta Cluster	Avg ROA
High	53%
Avg	220%
Low	136%
Tot	**156%**

Figure 6.14 – Out-of-sample analysis on the Dow Jones 30 list

By analyzing *Figure 6.14*, we can draw the following conclusions:

- Out of 30 symbols, 24 of them show profitable results.
- ROA varies from a negative value to a maximum of 7.56.
- Looking at the beta cluster analysis, we can see that the average beta cluster, representing stocks with volatility very similar to the index, shows an average ROA of 220%. On the contrary, both the low beta clusters and high beta clusters demonstrate lower performance.

The rationale behind the beta cluster analysis approach lies in the fact that stocks are not all equal due to their varying volatility. The volatility of individual stocks is a factor that allows us to group them and thus manage an optimization process that, on one hand, enables an out-of-sample analysis, and, on the other hand, is conducted on stocks that are not too diverse from each other.

Summary

In this chapter, we delved into a very important market pattern: trend pullback. Initially, we learned the rationale behind such market behavior and split it into three components: a trend, a pullback, and an impulse. After that, we wrote several algorithms to identify the different components and assemble them in various ways, akin to playing with bricks. Then, we conducted a multiple-sensitivity analysis to check strategy robustness.

As you may have noticed thus far, we focused solely on entry logic, leaving exits to only some holding periods. As you will better understand in the next chapter, dedicated to risk management, the choice of building algorithms that separate entries from exits is not arbitrary but, rather, has a valid reason behind it.

7

Risk Management

All financial markets share the trait of unpredictability. Although there are techniques that attempt to estimate future price movements, the reality is that it is not possible to predict the future. The only thing that can be predicted with absolute certainty is how much one is willing to put risk on each trade or the entire trading project. This fundamental principle of **risk management** is crucial for any trader, whether beginner or experienced.

You might have noticed that in all the algorithms we have written so far, we have mainly focused on entries. That is, we have created algorithms to identify trends, reversals, or trend pullbacks to open trades, but all trades have been managed with a simple holding period. This approach has helped us better focus on entry patterns, but now it is time to complete the development of our strategies by starting to manage the risk associated with trading.

Managing risk in trading strategies has two aspects: handling exits (or money management) and position sizing.

The purpose of this chapter is to get you to start thinking about algorithmic trading activity in terms of risk. By the end of this chapter, you will be able to manage both exits and position sizes algorithmically to keep financial risk under control at all times.

In this chapter, we are going to learn how to handle the following algorithmically:

- Money management
- Position sizing

There are various definitions of money management and position sizing. To avoid any misunderstanding, in this book, we will refer to **money management** as techniques for exiting trades, while **position sizing** will refer to techniques for determining the entry size of trades. The purpose of both is to control and manage trading risk.

Money management

When there are no open positions on the market, the trader is more clear-headed and can analyze the available alternatives more objectively. However, everything changes when a position is open: whether the market moves in our favor or against us, it becomes incredibly difficult to maintain the same clarity we had before. Therefore, algorithmic exit management is crucial as it can provide the

trader with much-needed support at the most critical moments. Sometimes traders use algorithms just to exit positions because they know this is a very difficult task.

The main exit tools are the following:

- **Stop loss**: A stop loss is a predefined price level set by a trader at which an open position will be automatically closed to limit potential losses
- **Target**: Target refers to the price level at which a trader aims to close a position to realize a profit
- **Trailing stop**: A trailing stop is a type of stop loss order that moves as the market price moves in a favorable direction

There are different approaches to developing such exit tools. In the next section, we will explore three of them:

- Price-based exits
- Percent-based exits
- Volatility-based exits

Price-based exits

Price-based exit tools are represented in the same price unit. For example, if we trade **Apple Inc.** (**AAPL**) with a current price of 190 USD, we might set a stop loss at 3 USD per share or a target at 10 USD per share, and so on.

We can write price-based exits as follows:

```
If average(close,4) cross over average(close,20) then buy 100 shares next bar open;
var:Htrade(0),LTrade(0);
if mp=0 then begin HTrade=H;LTrade=L;End;
if H>HTrade then HTRade = H;
if mp = 1 then begin
    Sell ("sl")     next bar entryprice - 3 stop;
    Sell ("tgt")    next bar entryprice + 10 limit;
    Sell ("trail")  next bar HTrade - 5 stop;
End;
```

In this script, we used the cross of **simple moving average** (**SMA**) to open long trades on AAPL with 100 shares for each trade.

The `Sell ("sl")  next bar entryprice - 3 stop;` instruction sets a stop loss of 3 USD per share, meaning that any trade will be closed if the price goes below our entry price minus 3 USD.

The `Sell ("tgt")  next bar entryprice + 10 limit;` instruction sets a target of 10 USD per share, meaning that any trade will be closed if the price goes above our entry price plus 10 USD.

The `HTrade` variable represents the highest price since we opened the trade, and the `Sell ("trail") next bar HTrade - 5 stop;` instruction sets a trailing stop of 5 USD per share, meaning that any trade will be closed if the price goes 5 USD below the highest price since the trade was opened.

> ***NOTE***
>
> *Please note that we used stop orders for the stop loss and trailing stop and a limit order for the target. This means that we expect to exit positions any time during the day, and in real-time operations, such orders will be sent to our broker as pending orders.*

In *Figure 7.1*, you can see how the price-based exit algorithms look in an AAPL chart.

Figure 7.1 – Price-based exits

Reading the chart in *Figure 7.1* from left to right, you can observe the following:

- A trailing stop is executed at a distance of 5 USD from the highest price after we entered the trade (which is circled)
- A target is executed 10 USD above the entry price
- A stop loss is executed 3 USD below the entry price

In *Figure 7.2*, you can see the related TradeStation trade list:

Type	Date/Time	Signal	Price	Roll Over Amt ...	Profit/Loss
Buy	10/10/2023	Buy	$178.10	$0.00	100
Sell	16/10/2023	trail	$176.75		($135.00)
Buy	07/11/2023	Buy	$179.18	$0.00	100
Sell	15/11/2023	tgt	$189.18		$1,000.00
Buy	23/01/2024	Buy	$195.02	$0.00	100
Sell	26/01/2024	sl	$192.02		($300.00)

Figure 7.2 – Price-based exits: trade list

Looking at the trade list in *Figure 7.2*, you can track the three trades. So, you can note that the first trade closed in trailing losses of $135, the second trade closed at a target with a profit of $1,000 (10 USD times 100 shares), and the third trade closed in the stop loss showing a loss of $300 (3 USD times 100 shares).

Percent-based exits

Many traders prefer managing risk as a percentage because price-based exits may not be suitable when trading baskets of stocks with different prices. For example, a stop loss of 3 USD per share equals 1.5% for AAPL, which trades at 190 USD, but it equals 10% for **Intel Corp.** (**INTC**), which trades at around 30 USD. Applying the same 3 USD stop to INTC would result in significantly higher risk compared to trading AAPL. To maintain the same risk percentage level, we can write the following script:

```
if mp = 1 then begin
    Sell ("sl")     next bar entryprice *(1-2*0.01) stop;
    Sell ("tgt")    next bar entryprice *(1+5*0.01) limit;
    Sell ("trail")  next bar HTrade - HTrade*3*0.01 stop;
End;
```

In this code, we can see the following:

- The **`entryprice *(1-2*0.01)`** formula calculates a stop level price 2% below the entry price
- The **`entryprice *(1+5*0.01)`** formula calculates a target price 5% above the entry price
- The **`HTrade - HTrade*3*0.01`** formula calculates a trailing price 3% below the highest price since we opened the trade

In *Figure 7.3*, we can see the trade signals on the AAPL chart:

Figure 7.3 – Percent-based exits

Reading the chart in *Figure 7.3* from left to right, you can observe the following:

- A trailing stop is executed at a distance of 3% from the highest price after we entered the trade (which is circled)
- A 5% target
- A 3% stop loss

In *Figure 7.4*, you can see the related TradeStation trade list:

Type	Date/Time	Signal	Price	Roll Over Amt ...	Profit/Loss
Buy	10/10/2023	Buy	$178.10	$0.00	100
Sell	16/10/2023	trail	$176.75		($135.00)
Buy	07/11/2023	Buy	$179.18	$0.00	100
Sell	15/11/2023	tgt	$188.14		$896.00
Buy	23/01/2024	Buy	$195.02	$0.00	100
Sell	29/01/2024	sl	$191.11		($391.00)

Figure 7.4 – Percent-based trade list

Looking at the trade list in *Figure 7.4*, you can track the three trades. Note that the first trade closed with a trailing loss of $135 (3% from the highest price after we entered the trade), the second trade closed at the target with a profit of $896 (5% of the entry price), and the third trade closed at the stop loss, showing a loss of $391 (3% of the entry price).

We can extend the same percentage-based logic to other stocks, even if they have completely different prices. However, as you will see in the coming section, this is not enough to properly manage risk due to volatility.

Volatility-based exits

As we learned in the previous chapters, volatility is a crucial aspect of financial markets. **Volatility** refers to the degree of variation in the price of a financial instrument over time. It is often measured by the standard deviation or variance of returns or prices. High volatility means the price of the asset can change dramatically over a short period in either direction, while low volatility means that the price does not fluctuate dramatically and tends to be more stable.

In this chapter, we are going to measure volatility with the **average true range** (**ATR**) and **average true range percent** (**ATRpct**).

We can write the following:

```
var:ATR(0),ATRPct(0);
ATR    = AvgTrueRange(200);
AtrPct = ATR / Close;
```

```
plot1(ATR,"ATR");
Plot2(Atrpct,"ATR%");
```

As you see, we used TradeStation's built-in **`AvgTrueRange`** function to calculate the ATR for the last 200 days.

The **`AtrPct = ATR / Close;`** formula calculates the ATR as a percent of the price.

The ATRpct is a better measure of volatility as it allows for the comparison of stocks with different prices.

In *Figure 7.5*, you can see the ATRpct indicator applied to the entire Dow Jones 30 list:

RadarScreen - Page 1

Timeframe | Studies | Data | Settings

Symbol	Description		Chapter7 ATRpct		+
	Name	Price	ATR	ATR%	
INTC	Intel Corp	31.42	1.09	3.5%	
BA	Boeing Co	186.28	4.73	2.5%	
NKE	Nike Inc Cl B	92.50	1.79	1.9%	
CRM	Salesforce Inc	283.82	5.02	1.8%	
AMZN	Amazon.com Inc	183.13	3.24	1.8%	
DIS	Disney (Walt) Co	103.02	1.82	1.8%	
CAT	Caterpillar Inc	355.94	6.03	1.7%	
AMGN	Amgen Inc	309.41	5.24	1.7%	
DOW	Dow Inc	57.60	0.96	1.7%	
CVX	Chevron Corporation	157.57	2.56	1.6%	
MMM	3M Co	101.49	1.63	1.6%	
UNH	UnitedHealth Grp Inc	521.35	8.15	1.6%	
VZ	Verizon Communications	39.79	0.62	1.6%	
HD	Home Depot Inc	330.59	5.11	1.5%	
AAPL	Apple Inc	190.90	2.87	1.5%	
GS(HB)	Goldman Sachs Grp	462.38	6.62	1.4%	
CSCO	Cisco Systems	47.43	0.67	1.4%	
MSFT	Microsoft Corp	430.52	6.09	1.4%	
HON	Honeywell Intl	202.80	2.83	1.4%	
MCD	McDonald's Corp	265.77	3.66	1.4%	
JNJ	Johnson & Johnson	153.50	2.10	1.4%	
IBM	Intl Business Machines Corp	173.69	2.34	1.3%	
MRK	Merck & Co	131.10	1.75	1.3%	
TRV	The Travelers Companies Inc	216.54	2.80	1.3%	
AXP	American Express Co	240.09	3.05	1.3%	
JPM	JPMorgan Chase & Co	198.31	2.48	1.2%	
V	Visa Inc	275.58	3.33	1.2%	
WMT	Walmart Inc	65.25	0.72	1.1%	
KO	Coca-Cola Co	63.00	0.69	1.1%	
PG	Procter & Gamble Co	167.81	1.82	1.1%	

Page 1

Figure 7.5 – ATR indicator

In *Figure 7.5*, you can see the Dow Jones 30 list sorted by ATRpct. At the top of the list is INTC with the highest ATRpct of 3.9%, while at the bottom is **Procter & Gamble Co** (**PG**) with the lowest ATRpct of 1.1%. As you can see, over the last 200 days, INTC has shown more than three times the volatility of PG. Applying the same percent-based exit to both of them will be too narrow for INTC and too loose for PG, leading to too many exits for the more volatile stock and too few exits for the less volatile one.

To solve this matter, we need to take volatility into account to manage exits.

Here's the script for volatility-based exits:

```
var:ATR(0);
ATR = AvgTrueRange(range,200);
if mp = 1 then begin
    Sell ("sl")     next bar entryprice - 2*ATR stop;
    Sell ("tgt")    next bar entryprice + 5*ATR limit;
    Sell ("trail")  next bar HTrade     - 3*ATR stop;
End;
```

In this code, the **`ATR = average(range,200);`** formula calculates the volatility in terms of the average range over the last 200 days. It will be a different value for different stocks, to make the exit management adaptive.

In *Figure 7.6*, you can see the volatility-based exit on AAPL charts:

Figure 7.6 – Volatility-based exits

Reading the chart in *Figure 7.6* from left to right, you can observe the following:

- A trailing stop is executed at a distance of 2 ATRs from the highest price after we entered the trade (which is circled)
- A 5 ATR target
- A 3 ATR stop loss

In *Figure 7.7*, you can see the related TradeStation trade list:

Type	Date/Time	Signal	Price	Roll Over Amt ...	Profit/Loss
Buy	10/10/2023	Buy	$178.10	$0.00	100
Sell	20/10/2023	trail	$173.49		($461.00)
Buy	07/11/2023	Buy	$179.18	$0.00	100
Sell	05/12/2023	tgt	$193.15		$1,397.00
Buy	23/01/2024	Buy	$195.02	$0.00	100
Sell	30/01/2024	sl	$189.50		($552.00)

Figure 7.7 – Volatility-based trade list

Looking at the trade list in *Figure 7.7*, you can track the three trades. So, you can note the following:

- The first trade closed with a tailing stop losing $461 (2 ATRs from the highest price after we entered the trade)
- The second trade is closed at target with a profit of $1,397 (5 ATRs above the entry price)
- The third trade closed in stop loss shows a loss of $552 (3 ATRs below the highest price after we entered the trade)

If we apply volatility-based exit logic, our trading risk will be adjusted to different historical volatility market conditions and the different assets' volatility. However, there is still something missing, especially when trading a portfolio of very different stocks in terms of volatility. Let's see why.

For example, imagine we trade a two-stock portfolio composed of the least volatile and the most volatile stock according to *Figure 7.5*: INTC and PG. Let's say we bought 10,000 USD of both and apply 1 ATR stop loss. The results are as follows:

- A 3.5% loss on a $10,000 investment in INTC, totaling $350
- A 1.1% loss on a $10,000 investment in PG, totaling $110

In simpler terms, the volatility-based approach alone isn't sufficient to offset the dollar risk when trading stocks with significantly different levels of volatility.

To solve such a matter, we need to look at position sizing.

Position sizing

In the previous section, we tested three different exit techniques to control trading risk. While each of the three techniques can work well on single assets, they cannot function as they did when trading a portfolio. A dollar stop is not suitable because different stocks have vastly different prices; a percent-based stop is not suitable because different stocks have different volatility levels, and a volatility-based stop increases the risk toward the most volatile stocks in terms of absolute dollar risk.

Let's try a different approach to break out of this negative loop: **equal dollar risk sizing**.

Equal dollar risk sizing is a risk management strategy where you allocate the same amount of money, or dollar amount, to each trade or investment. This means that regardless of the price of the asset, or its volatility, you are risking the same fixed dollar amount on each trade.

In my experience, I've found that exit management techniques may require different logic depending on whether they are based on technical levels or not. Let's delve into this crucial concept further.

In the previous section, we tested some exit logic with trading strategies not requiring *technical-level exits*. However, in reality, trading strategies often require detecting exit points according to historical prices or patterns. In such cases, the exit levels will not be calculated by a risk management formula but by a technical algorithm.

> ***NOTE***
>
> *By technical-level exits, we mean exit points calculated by technical algorithms, such as a previous low for a stop loss or a trailing stop, or a previous high for a target.*

Before proceeding with the development of position-sizing algorithms, it is advisable to frame the problem scientifically. In the next section, we will develop some very simple systems of equations that, starting from the **profit and loss** (**P&L**) formula, will allow us to obtain the formula for calculating the correct sizes for risk management.

We are going to learn about the following:

- The P&L equation
- Equal dollar risk technique without technical exit levels
- Incorporating technical exits in equal dollar risk formulas

The P&L equation

The P&L equation is as follows:

```
P&L = Shares x (ExitPrice–EntryPrice)
```

Here, we have the following:

- **`P&L`** is the total P&L for the single trade (no commission included)
- **`Shares`** is the total number of shares bought
- **`EntryPrice`** is the price we bought the stock at
- **`ExitPrice`** is the price we sold the stock at

> ***NOTE***

We are not including costs such as commissions and slippage because this is beyond the purpose of this section.

Equal dollar risk technique without technical exit levels

In this section, we are going to do the following:

- Obtain the equal risk dollar size formula from the P&L function
- Write the equal dollar risk algorithm in EasyLanguage
- Perform some backtestings

Equal dollar risk formula without technical exits

Starting with the P&L equation, we can obtain the following equal dollar risk size formula:

```
Shares <=   DollarRisk / (NumAtr x ATR)
In Figure 7.8 you can see the mathematical steps to obtain the formula.
```

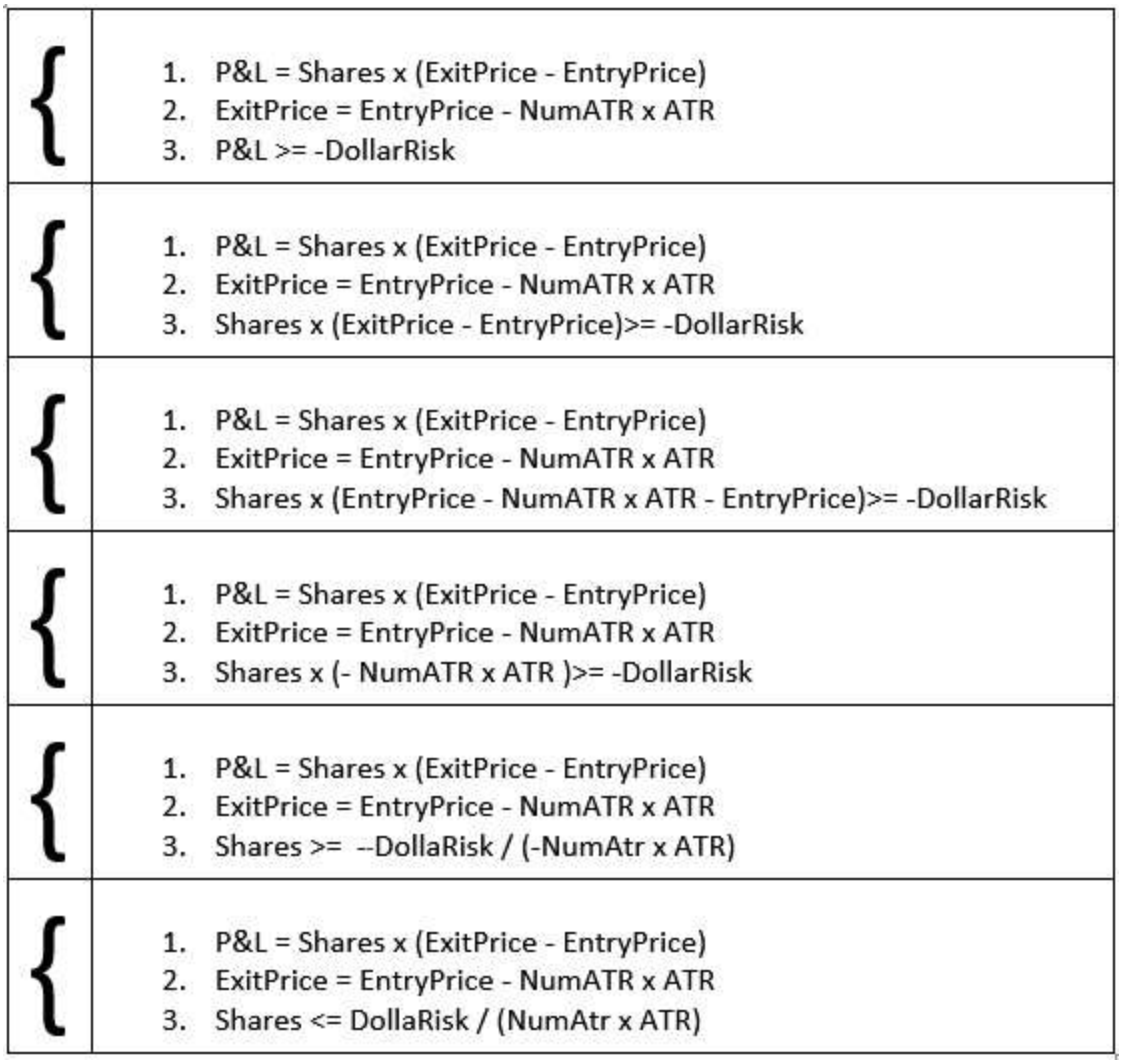

1. P&L = Shares x (ExitPrice - EntryPrice)
2. ExitPrice = EntryPrice - NumATR x ATR
3. P&L >= -DollarRisk
1. P&L = Shares x (ExitPrice - EntryPrice)
2. ExitPrice = EntryPrice - NumATR x ATR
3. Shares x (ExitPrice - EntryPrice)>= -DollarRisk
1. P&L = Shares x (ExitPrice - EntryPrice)
2. ExitPrice = EntryPrice - NumATR x ATR
3. Shares x (EntryPrice - NumATR x ATR - EntryPrice)>= -DollarRisk
1. P&L = Shares x (ExitPrice - EntryPrice)
2. ExitPrice = EntryPrice - NumATR x ATR
3. Shares x (- NumATR x ATR)>= -DollarRisk
1. P&L = Shares x (ExitPrice - EntryPrice)
2. ExitPrice = EntryPrice - NumATR x ATR
3. Shares >= --DollaRisk / (-NumAtr x ATR)
1. P&L = Shares x (ExitPrice - EntryPrice)
2. ExitPrice = EntryPrice - NumATR x ATR
3. Shares <= DollaRisk / (NumAtr x ATR)

Figure 7.8 – Equal dollar risk equations system

Let's look at a practical example. Let's say we want to maintain a maximum loss of 1,000 USD trading both INTC and PG. By applying the equal dollar risk size formula and assuming `NumAtr=1`, we can calculate the stocks' sizes in *Table 7.1*:

Stocks	**Price**	**ATR**	**Shares**	**Total Investment (USD)**
INTC	31.42	1.01	Shares <= 1,000 / 1.01 = 917	917 x 31.42 = 28,812
PG	167.81	1.67.81	Shares <= 1,000 / 1.80 = 549	549 x 167.81 = 92,128

Table 7.1 – Equal dollar risk sizes for INTC and PG

As you can see in *Table 7.1*, if we want to maintain the same dollar risk, we need to buy a maximum of 549 shares of PG, and no more than 917 shares of INTC. The total investment on PG is 92,128

USD, while on INTC, which is more volatile, we can invest up to 28,812 USD.

Let’s now move on to how to write such formulas in EasyLanguage, to automate such an important process.

Writing the equal dollar risk formula algorithm

We can write the strategy script as follows:

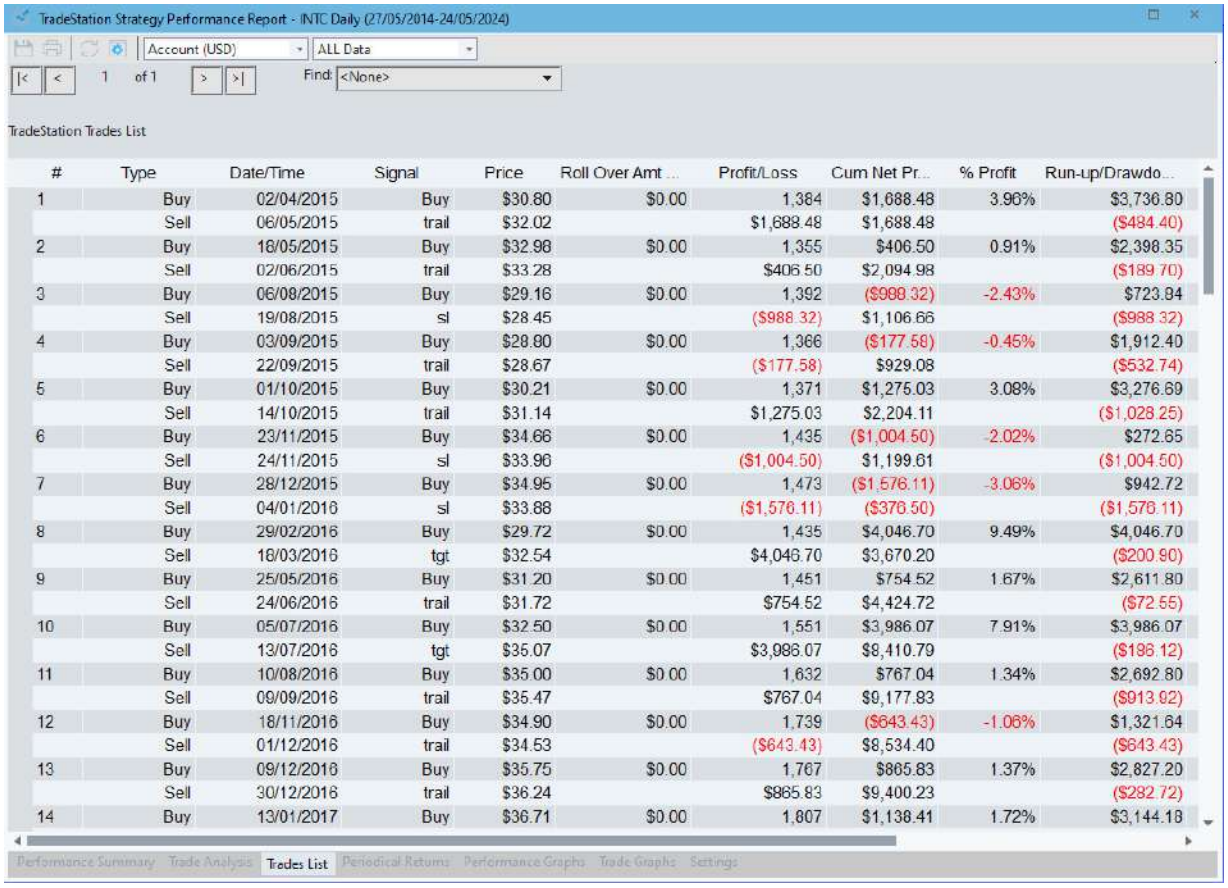
```
input:AtrLen(200),StopATRs(1),TgtATRs(2),TrailATRs(2),DollarRisk(1000);
var:ATR(0);
ATR=avgtruerange(ATRLen);
var: qty(0);
Qty = DollarRisk / (StopATRs*ATR);
If average(close,4) cross over average(close,20) then buy qty shares next bar open;
var:Htrade(0),LTrade(0);
if mp=0 then begin HTrade=H;LTrade=L;End;
if H>HTrade then HTRade = H;
if L<LTrade then LTrade = L;
if mp = 1 then begin
    Sell ("sl")     next bar entryprice - StopATRs*ATR stop;
    Sell ("tgt")    next bar entryprice + TgtATRs*ATR limit;
    Sell ("trail")  next bar HTrade     - TrailATRs*ATR stop;
End;
```

The **`Qty = DollarRisk / (StopATRs*ATR);`** instruction calculates sizes according to *Table 7.1*.

The **`Sell ("sl") next bar entryprice - StopATRs*ATR stop;`** instruction executes a stop loss order at a variable distance from the entry price, depending on the specific asset’s volatility.

The same logic applies to the target and trailing stop. Let’s see how the backtesting on two very different assets such as INTC and PG works.

In *Figure 7.9*, you can see the TradeStation trade list for INTC.



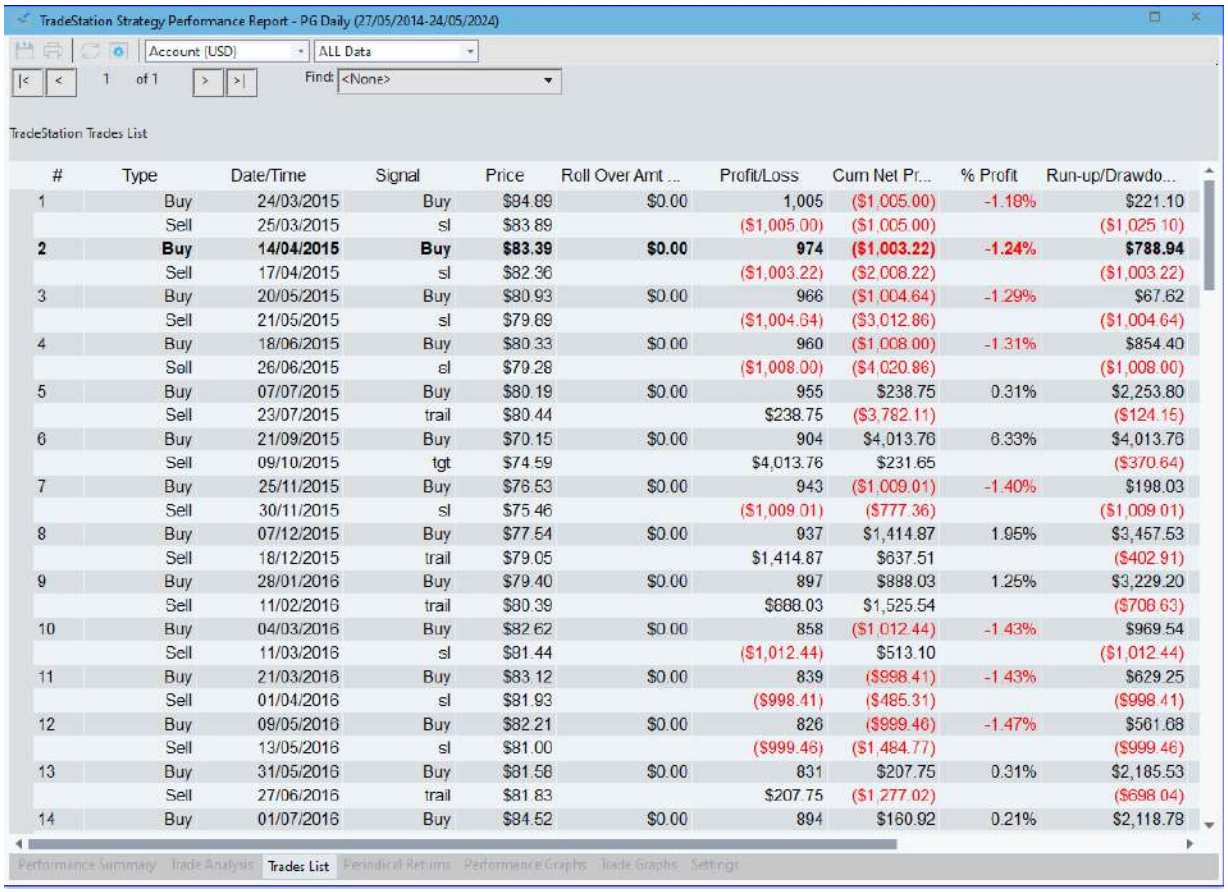
#	Type	Date/Time	Signal	Price	Roll Over Amt ...	Profit/Loss	Cum Net Pr...	% Profit	Run-up/Drawdo...
1	Buy	02/04/2015	Buy	$30.80	$0.00	1,384	$1,688.48	3.96%	$3,736.80
	Sell	06/05/2015	trail	$32.02		$1,688.48	$1,688.48		($484.40)
2	Buy	18/05/2015	Buy	$32.98	$0.00	1,355	$406.50	0.91%	$2,398.35
	Sell	02/06/2015	trail	$33.28		$406.50	$2,094.98		($189.70)
3	Buy	06/08/2015	Buy	$29.16	$0.00	1,392	($988.32)	-2.43%	$723.84
	Sell	19/08/2015	sl	$28.45		($988.32)	$1,106.66		($988.32)
4	Buy	03/09/2015	Buy	$28.80	$0.00	1,366	($177.58)	-0.45%	$1,912.40
	Sell	22/09/2015	trail	$28.67		($177.58)	$929.08		($532.74)
5	Buy	01/10/2015	Buy	$30.21	$0.00	1,371	$1,275.03	3.08%	$3,276.69
	Sell	14/10/2015	trail	$31.14		$1,275.03	$2,204.11		($1,028.25)
6	Buy	23/11/2015	Buy	$34.66	$0.00	1,435	($1,004.50)	-2.02%	$272.65
	Sell	24/11/2015	sl	$33.96		($1,004.50)	$1,199.61		($1,004.50)
7	Buy	28/12/2015	Buy	$34.95	$0.00	1,473	($1,576.11)	-3.06%	$942.72
	Sell	04/01/2016	sl	$33.88		($1,576.11)	($376.50)		($1,576.11)
8	Buy	29/02/2016	Buy	$29.72	$0.00	1,435	$4,046.70	9.49%	$4,046.70
	Sell	18/03/2016	tgt	$32.54		$4,046.70	$3,670.20		($200.90)
9	Buy	25/05/2016	Buy	$31.20	$0.00	1,451	$754.52	1.67%	$2,611.80
	Sell	24/06/2016	trail	$31.72		$754.52	$4,424.72		($72.55)
10	Buy	05/07/2016	Buy	$32.50	$0.00	1,551	$3,986.07	7.91%	$3,986.07
	Sell	13/07/2016	tgt	$35.07		$3,986.07	$8,410.79		($196.12)
11	Buy	10/08/2016	Buy	$35.00	$0.00	1,632	$767.04	1.34%	$2,692.80
	Sell	09/09/2016	trail	$35.47		$767.04	$9,177.83		($913.92)
12	Buy	18/11/2016	Buy	$34.90	$0.00	1,739	($643.43)	-1.06%	$1,321.64
	Sell	01/12/2016	trail	$34.53		($643.43)	$8,534.40		($643.43)
13	Buy	09/12/2016	Buy	$35.75	$0.00	1,767	$865.83	1.37%	$2,827.20
	Sell	30/12/2016	trail	$36.24		$865.83	$9,400.23		($282.72)
14	Buy	13/01/2017	Buy	$36.71	$0.00	1,807	$1,138.41	1.72%	$3,144.18

Figure 7.9 – TradeStation trade list with equal dollar risk formula INTC

As you see in *Figure 7.9*, under the **`Profit/Loss`** column, there is just one trade with a loss higher than 1,000 USD, due to a gap open on December 28, 2015.

> ***NOTE***
>
> *A **gap** refers to a situation where a stock's price opens significantly higher or lower than its previous closing price, leaving a blank space or "gap" on a price chart. This phenomenon occurs due to various reasons, such as after-hours news, earnings announcements, or other significant events that impact investor sentiment.*

In *Figure 7.10*, we can see the TradeStation trade list on PG:



#	Type	Date/Time	Signal	Price	Roll Over Amt ...	Profit/Loss	Cum Net Pr...	% Profit	Run-up/Drawdo...
1	Buy	24/03/2015	Buy	$84.89	$0.00	1,005	($1,005.00)	-1.18%	$221.10
	Sell	25/03/2015	sl	$83.89		($1,005.00)	($1,005.00)		($1,025.10)
2	**Buy**	**14/04/2015**	**Buy**	**$83.39**	**$0.00**	**974**	**($1,003.22)**	**-1.24%**	**$788.94**
	Sell	17/04/2015	sl	$82.36		($1,003.22)	($2,008.22)		($1,003.22)
3	Buy	20/05/2015	Buy	$80.93	$0.00	966	($1,004.64)	-1.29%	$67.62
	Sell	21/05/2015	sl	$79.89		($1,004.64)	($3,012.86)		($1,004.64)
4	Buy	18/06/2015	Buy	$80.33	$0.00	960	($1,008.00)	-1.31%	$854.40
	Sell	26/06/2015	sl	$79.28		($1,008.00)	($4,020.86)		($1,008.00)
5	Buy	07/07/2015	Buy	$80.19	$0.00	955	$238.75	0.31%	$2,253.80
	Sell	23/07/2015	trail	$80.44		$238.75	($3,782.11)		($124.15)
6	Buy	21/09/2015	Buy	$70.15	$0.00	904	$4,013.76	6.33%	$4,013.76
	Sell	09/10/2015	tgt	$74.59		$4,013.76	$231.65		($370.64)
7	Buy	25/11/2015	Buy	$76.53	$0.00	943	($1,009.01)	-1.40%	$198.03
	Sell	30/11/2015	sl	$75.46		($1,009.01)	($777.36)		($1,009.01)
8	Buy	07/12/2015	Buy	$77.54	$0.00	937	$1,414.87	1.95%	$3,457.53
	Sell	18/12/2015	trail	$79.05		$1,414.87	$637.51		($402.91)
9	Buy	28/01/2016	Buy	$79.40	$0.00	897	$888.03	1.25%	$3,229.20
	Sell	11/02/2016	trail	$80.39		$888.03	$1,525.54		($708.63)
10	Buy	04/03/2016	Buy	$82.62	$0.00	858	($1,012.44)	-1.43%	$969.54
	Sell	11/03/2016	sl	$81.44		($1,012.44)	$513.10		($1,012.44)
11	Buy	21/03/2016	Buy	$83.12	$0.00	839	($998.41)	-1.43%	$629.25
	Sell	01/04/2016	sl	$81.93		($998.41)	($485.31)		($998.41)
12	Buy	09/05/2016	Buy	$82.21	$0.00	826	($999.46)	-1.47%	$561.68
	Sell	13/05/2016	sl	$81.00		($999.46)	($1,484.77)		($999.46)
13	Buy	31/05/2016	Buy	$81.58	$0.00	831	$207.75	0.31%	$2,185.53
	Sell	27/06/2016	trail	$81.83		$207.75	($1,277.02)		($698.04)
14	Buy	01/07/2016	Buy	$84.52	$0.00	894	$160.92	0.21%	$2,118.78

Figure 7.10 – TradeStation trade list with equal dollar risk formula PG

As you can see in *Figure 7.10*, under the **`Profit/Loss`** column, there are few trades without a loss higher than 1,000 USD. Again, such differences are due to gaps.

In this section, we uncovered a technique to manage risk on stock portfolios when the trading strategy does not require technical exits. In the next section, we will move on to how to deal with position sizing, taking technical exit levels into account.

Incorporating technical exits in equal dollar risk formulas

If our exit strategy involves technical levels, the formula for calculating the size changes, as the exit levels will be determined by technical indicators.

Let’s go through an example. Suppose we want to enter a position when a fast SMA crosses above a slow SMA and set a stop loss by closing the position if the price drops below the lowest low of the past 5 days.

In this case, even though we don't know where the stop loss level will be when we open a trade, we already know the formula to calculate it.

Look at the following script:

```
If mp =0 and FastSMA cross above SlowSMA then begin
    entry next bar open;
    stoplevel = lowest(L,5);
End;
```

The **`stoplevel = lowest(L,5);`** instruction calculates the stop level at the same time the strategy opens the trade. This exit level could be anywhere below the entry price, depending on how the price moved in the last five days. Since we cannot choose the exit level, as the algorithm performs such a task, the only way to control the risk is by adjusting the position size.

In this section, we are going to do the following:

- Obtain the equal risk dollar size formula from the P&L function
- Writing the related algorithm in EasyLanguage
- Present some backtesting

Equal dollar risk formula with technical exits

Starting from the P&L equation, we can obtain the following equal dollar risk size formula:

`Shares < - DollaRisk / (ExitPrice – EntryPrice)`

As you can see, the formula is quite simple as a technical indicator will provide the **`ExitPrice`** value.

Writing the equal dollar risk algorithm in the presence of technical exits

Let's have a look at the following script, where we designed a strategy to maintain a maximum **`DollarRisk`** of 1,000 USD in the presence of technical exits based on the lowest lows of the last five days:

```
input:DollarRisk(1000);
var:StopLevel(0),qty(0);
If mp=0 and  average(close,4) cross over average(close,20) then begin
    StopLevel = Lowest(L,5);
    qty = -DollarRisk /(StopLevel-Close);
    buy qty shares this bar close;
End;
if mp = 1 then  begin
    value1=text_new(date,time,StopLevel,"-----------------");
    text_setstyle(value1,0,2);
    text_setcolor(value1,red);
End;
sell ("sl") next bar StopLevel stop;
if barssinceentry =  15 then sell next bar open;
```

The `qty = -DollarRisk /(StopLevel-Close);` instruction applies the formula of *Table 7.2* inside the algorithm *only when* the strategy opens a new position.

The `value1=text_new(date,time,StopLevel,"-----------------");` instruction draws a dotted line representing the technical stop exits, which is very helpful in visualizing the exit point on the chart.

Let's see an example with PG in *Figure 7.11*:

Figure 7.11 – Trades with an equal dollar risk size on PG

In *Figure 7.11*, you can see that there are two occasions where the fast SMA crosses over the slow SMA.

Reading the chart from left to right, the first time the strategy buys 265 shares corresponding to roughly 39,000 USD, while the second time the strategy buys 142 shares equaling 23,000 USD. The second trade, due to the higher distance from the technical stop level, shows a trade value 42% lower than the first trade.

The two horizontal red lines represent the technical stop levels. The algorithm levels the sizes according to the distance from the stop level and the entry price to keep the maximum loss under 1,000 USD.

If you apply this logic, no matter which stock you have in your portfolio, no matter what the stocks' prices and volatility are, you can control the trade risk by maintaining it (except for gaps) below 1,000 USD with a technical stop loss level at the last 5 days' low.

Summary

In this chapter, we went through one of the most important aspects of algorithmic trading: risk management. We learned that managing risk involves two main components: managing exits and sizing positions, which is especially important for stock portfolio traders. Then, we delved into how to build exit trading algorithms according to different market logic and how to manage position sizes in both the presence and absence of technical exits. Keeping risk under control greatly increases traders' confidence, and managing automated exits gives them the freedom to focus on the next entry.

Although these techniques allow for adaptive risk management on each trade based on both historical volatility and the varying volatility of assets, there remains a partial problem with gap risk. In the next chapter, we will explore 24-hour trading assets such as Futures and Forex, where the gap risk will be mitigated.

8

Futures and Forex Algorithmic Trading

So far, we have seen how to build trading algorithms for stock markets. This was a deliberate choice since the stock market is the simplest to start with. Buying a stock means buying a small part of a company; if the company sells its products, its customers are happy, and it manages to increase its profits and raise the value of its shares. Conversely, the value of its shares is destined to fall, sometimes sharply. As you may have noticed, trading strategies in the stock market are generally long, as it is very difficult for short strategies to work, except in the short term.

In this chapter, we will apply many of the concepts we have learned so far to the futures and forex markets. By the end of this chapter, we will have a better understanding of the functioning of the futures and forex markets and how to deal with such topics through TradeStation. The objective of this chapter is for you to learn how to manage algorithms suitable for these particular markets, which, as you will see, will be used in both long and short strategies.

In this chapter, we are going to delve into the following topics:

- Algorithmic trading for futures markets
- Forex algorithmic trading for forex markets

This book provides you with a method for building algorithmic strategies and includes some practical examples. However, there are dozens of trading approaches that come from the practical experience of hundreds of traders who faced the markets before you. My advice is to remain curious about new trading approaches, even if they don’t originate from the algorithmic world. If I’ve done my job well in this book, you will have the know-how to transform any trading idea into an algorithm and to determine, through backtesting processes, whether these techniques can be successful or whether they are simply smoke and mirrors.

Gaining an in-depth understanding of the futures and forex markets will broaden your knowledge beyond the stock markets, giving you the opportunity to choose the markets that are most suited to your characteristics.

Let’s start with futures.

Algorithmic trading for futures markets

Futures contracts have ancient origins, dating back centuries when traders began entering agreements for the future delivery of agricultural commodities such as wheat, livestock, and other

basic goods. These early contracts were often privately negotiated between interested parties.

However, the modern concept of standardized futures contracts traded on an exchange date to Europe and America during the 19th and 20th centuries. Commodity exchanges developed in major financial centers such as Chicago, New York, and London, offering a wide range of futures contracts on agricultural commodities, metals, energy, and other goods.

The standardization of futures contracts, which included specifications such as contract sizes, delivery dates, and methods, helped to increase the liquidity and efficiency of futures markets, allowing a broader range of participants to access these financial instruments.

Today, futures contracts are traded on a wide range of financial assets, including stocks, stock indices, bonds, currencies, interest rates, and commodities. They are used by a variety of market participants, including producers, consumers, institutional investors, traders, and speculators, for the purposes of hedging, speculation, and risk management.

TradeStation offers more than 100 futures contracts, as classified in *Table 8.1*. You can find the full list at https://www.tradestation.com/pricing/futures-margin-requirements/.

#	Category	Main Futures Traded
1	Indexes	E-mini S&P 500, E-mini Nasdaq 100, E-mini Dow Jones
2	Eurex	EURO STOXX 50, Dax, Bund
3	Euronext Liffe	London FTSE 100 Index Futures, Long Gilt Futures
4	Currencies	Australian Dollar, Euro Currency, Japanese Yen, British Pound
5	Interest Rates	2 years Note, 5 years Note, 10 years Note, 20 years Note, 30 years Note
6	Metals	Gold, Silver, Copper, Palladium
7	Energies	Crude Oil, Natural Gas, Heating Oil
8	Agriculture	Wheat, Corn, Soybeans
9	Meats	Lean Hogs, Live Cattle, Feeder Cattle
10	Softs	Cotton, Coffe, Cocoa, Sugar

Table 8.1 – The TradeStation futures list

As you can see in *Table 8.1*, through futures contracts, it is possible to trade in a wide range of markets. In contrast to stock trading, futures trading is more complex and requires some basic information before delving into algorithm development.

In this section, we are going to learn about the following:

- The basic concepts of futures
- How to write a time breakout strategy

The basic concepts of futures

To understand how futures trading works, we need to learn some basic concepts related to the following:

- Expiration dates and rollovers
- Continuous contracts
- Margin trading
- Futures sessions

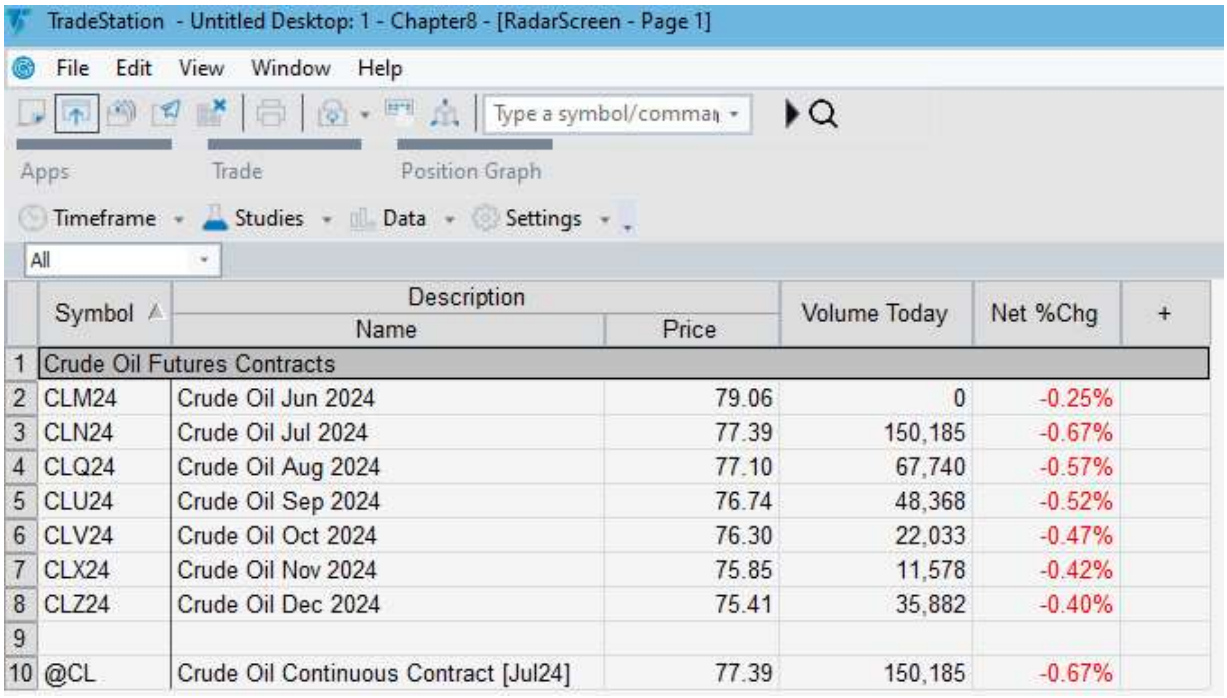
Expiration dates and rollovers

Any futures contract has an **expiration date**. The expiration date of a futures contract refers to when the contract ceases to exist. It is the last day on which the contract can be traded before it is settled. For physically settled contracts, the expiration date is also the date on which the underlying asset must be delivered or received, depending on the terms of the contract. For those not interested in physical deliveries, it is important to remember such a date and don't forget to roll over contracts. This means closing out a position in a futures contract that is nearing its expiration date and simultaneously opening a new position in a futures contract with a later expiration date.

Different futures contracts can have different expiration dates. For example, a crude oil future has 12 monthly expirations, while futures on indices only have 4 (1 per quarter). Before trading any futures contract, you should know what the front contract is.

> ***NOTE***
>
> *A **front contract**, also known as a **front-month contract**, is a futures contract that has the earliest expiration date. It is the contract that is closest to maturity among all the futures contracts available for trading on a particular underlying asset.*

It is generally advisable to trade the front contracts that are the most liquid. In *Figure 8.1*, you can see some crude oil futures contracts with consecutive expiration dates.



	Symbol	Description		Volume Today	Net %Chg	+
		Name	Price			
1	Crude Oil Futures Contracts					
2	CLM24	Crude Oil Jun 2024	79.06	0	-0.25%	
3	CLN24	Crude Oil Jul 2024	77.39	150,185	-0.67%	
4	CLQ24	Crude Oil Aug 2024	77.10	67,740	-0.57%	
5	CLU24	Crude Oil Sep 2024	76.74	48,368	-0.52%	
6	CLV24	Crude Oil Oct 2024	76.30	22,033	-0.47%	
7	CLX24	Crude Oil Nov 2024	75.85	11,578	-0.42%	
8	CLZ24	Crude Oil Dec 2024	75.41	35,882	-0.40%	
9						
10	@CL	Crude Oil Continuous Contract [Jul24]	77.39	150,185	-0.67%	

Figure 8.1 – Crude oil expiration dates

In *Figure 8.1*, you can see a TradeStation RadarScreen screenshot with seven consecutive expiration dates, ranging from June 2024 (row 2) to December 2024 (row 8). If you look at the **Volume Today** column, you can see that the June contract has no volume, meaning that it is expired, while the July contract has the highest volume. This means that July, at the time of writing, is the front contract.

Next, let's see what a continuous contract is.

Continuous contracts

A **continuous futures contract** combines the price data of multiple individual futures contracts into a single, uninterrupted price series. This synthetic contract provides a long-term view of price movements for an underlying asset, without the need to switch between individual contracts as they expire.

Futures continuous contracts play a crucial role in backtesting because they provide a seamless historical price series by rolling over individual futures contracts. This removes gaps and discontinuities that occur when contracts expire and new ones begin, ensuring a consistent dataset for analysis.

TradeStation offers several alternatives to build continuous contracts, and, depending on your need, you can customize this process.

From the menu on top of the chart, go to **Data** | **Edit Symbol** | **Settings** | **Lookup** | **Custom Future**.

In *Figure 8.2*, you can see the TradeStation **Custom Futures** window for CL (the crude oil future):

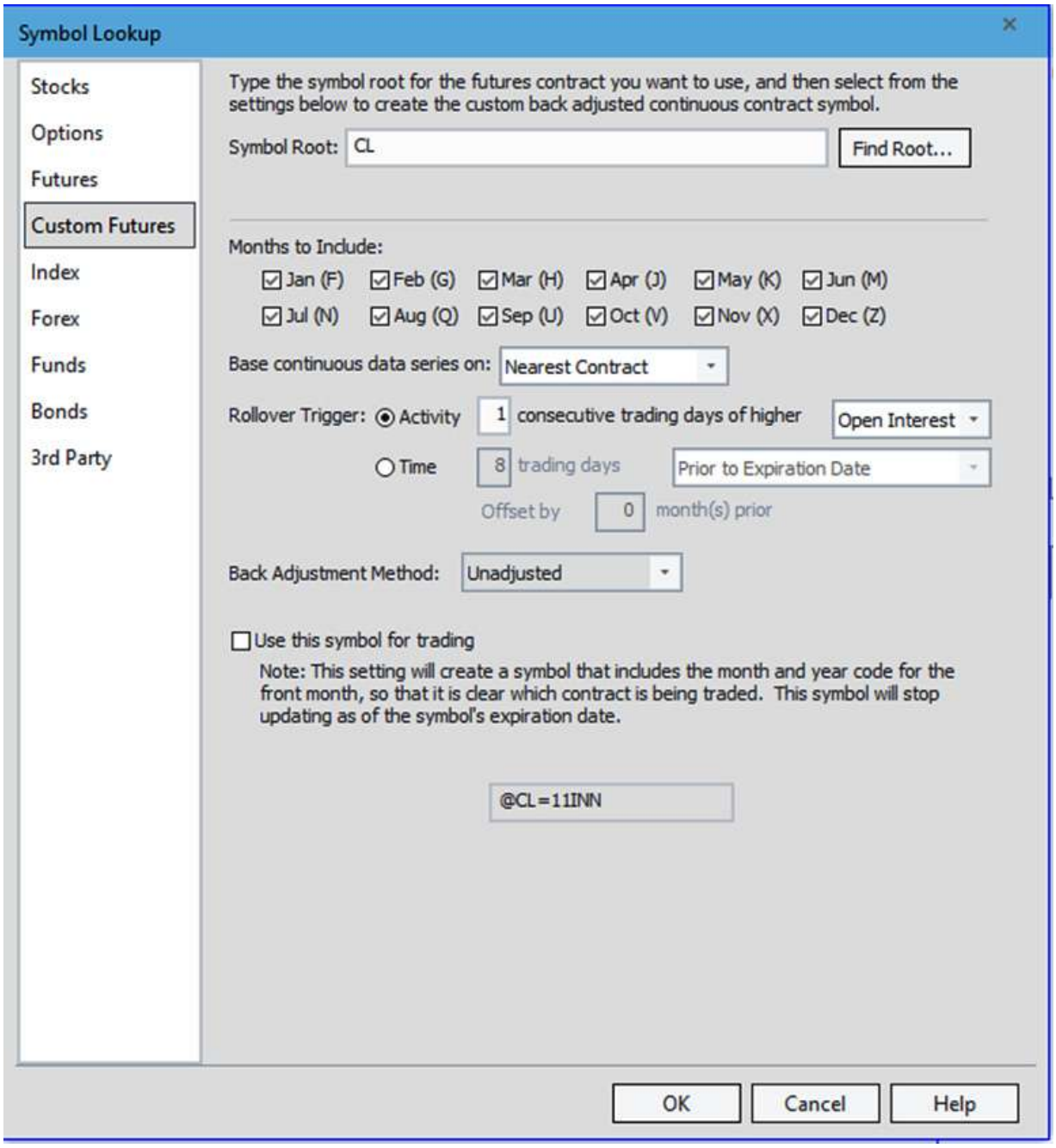

Symbol Lookup
Stocks
Options
Futures
Custom Futures
Index
Forex
Funds
Bonds
3rd Party
Type the symbol root for the futures contract you want to use, and then select from the settings below to create the custom back adjusted continuous contract symbol.
Symbol Root: CL
Find Root...
Months to Include:
Jan (F) Feb (G) Mar (H) Apr (J) May (K) Jun (M)
Jul (N) Aug (Q) Sep (U) Oct (V) Nov (X) Dec (Z)
Base continuous data series on: Nearest Contract
Rollover Trigger: Activity 1 consecutive trading days of higher Open Interest
Time 8 trading days Prior to Expiration Date
Offset by 0 month(s) prior
Back Adjustment Method: Unadjusted
Use this symbol for trading
Note: This setting will create a symbol that includes the month and year code for the front month, so that it is clear which contract is being traded. This symbol will stop updating as of the symbol's expiration date.
@CL=11INN
OK
Cancel
Help

Figure 8.2 – Customizing the Continuous Futures window for CL

In the window in *Figure 8.2*, you can create a custom continuous future. TradeStation offers a lot of different ways to create customization, but in my experience, there are two main customizations to use:

- **@CL**: This is the basic symbol for an adjusted continuous contract
- **@CL=11INN**: This is the symbol for an unadjusted continuous contract

In *Figure 8.3*, you can see the two crude oil symbols on the same chart:

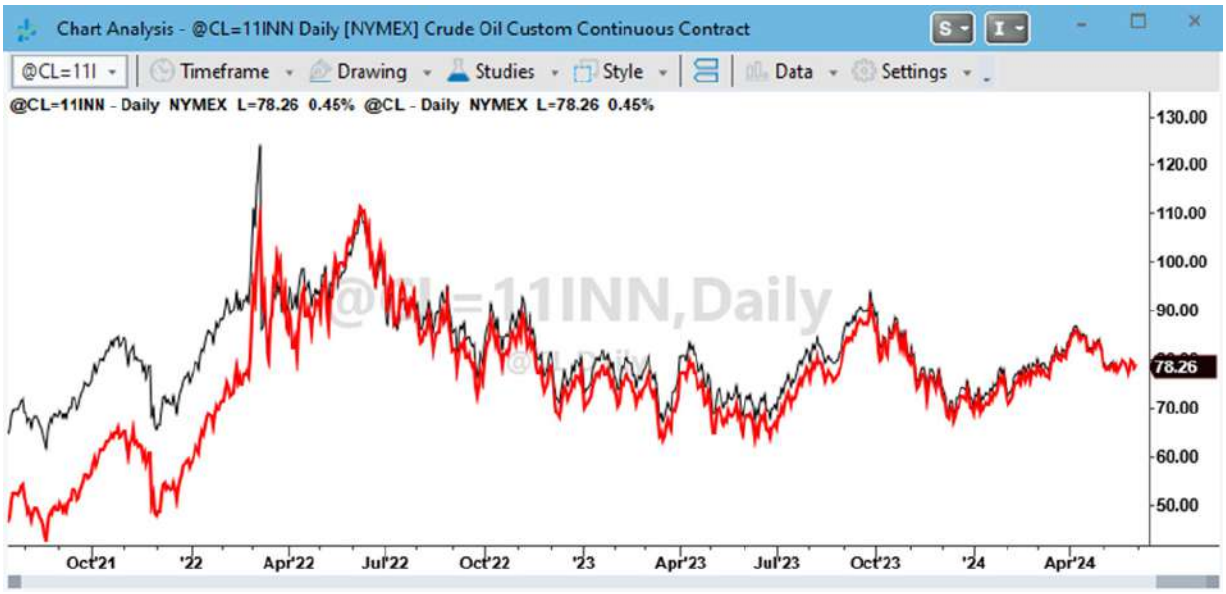


Figure 8.3 – The crude oil future – an adjusted Contract versus an unadjusted contract

In *Figure 8.3*, you can see the unadjusted crude oil future (the thin black line) and the adjusted crude oil future (the thick red line). Note how, on the right of the chart, the two prices overlap, while on the left of the chart, they show several dollars' difference.

As a rule of thumb, for a multi-day strategy, it is advisable to use the adjusted **@CL** symbol; alternatively, for an intraday strategy, it is advisable to double-check backtesting results on **@CL=11INN**, which is closer to real-world historical trade prices.

No matter which approach you choose for the backtesting, in real life, you have to remember to roll over.

> ***NOTE***
>
> *A **rollover** in the context of futures trading refers to the process of closing out an existing futures contract position near its expiration date and, simultaneously, opening a new position in a contract with a later expiration date. Traders often use rollovers to maintain exposure to the market while avoiding the need to take physical delivery or settle an expiring contract.*

Another important aspect of futures trading is the **margin**.

Margin trading

Margin trading in futures refers to the practice of trading futures contracts using borrowed funds, where traders are required to deposit a fraction of the total contract value as collateral. Brokers determine the margins for each futures contract based on its volatility characteristics.

Any time you trade futures, you need to focus on margin to avoid a margin call from the broker.

> ***NOTE***
>
> *A **margin call** is a demand from a broker or exchange for an investor to deposit additional funds or securities into their account to meet margin requirements. It occurs when the value of securities held in a margin account falls below a certain threshold, known as the **maintenance margin level**.*

Futures sessions

Different futures contracts, depending on the exchange where they are traded, may have different trading sessions. To check futures trading hours, you can go to an exchange website such as CME Group (https://www.cmegroup.com/), or, from the TradeStation chart menu, you can go to **Data | Edit Symbol | Properties | Sessions**.

A window like the one shown in *Figure 8.4* will appear:

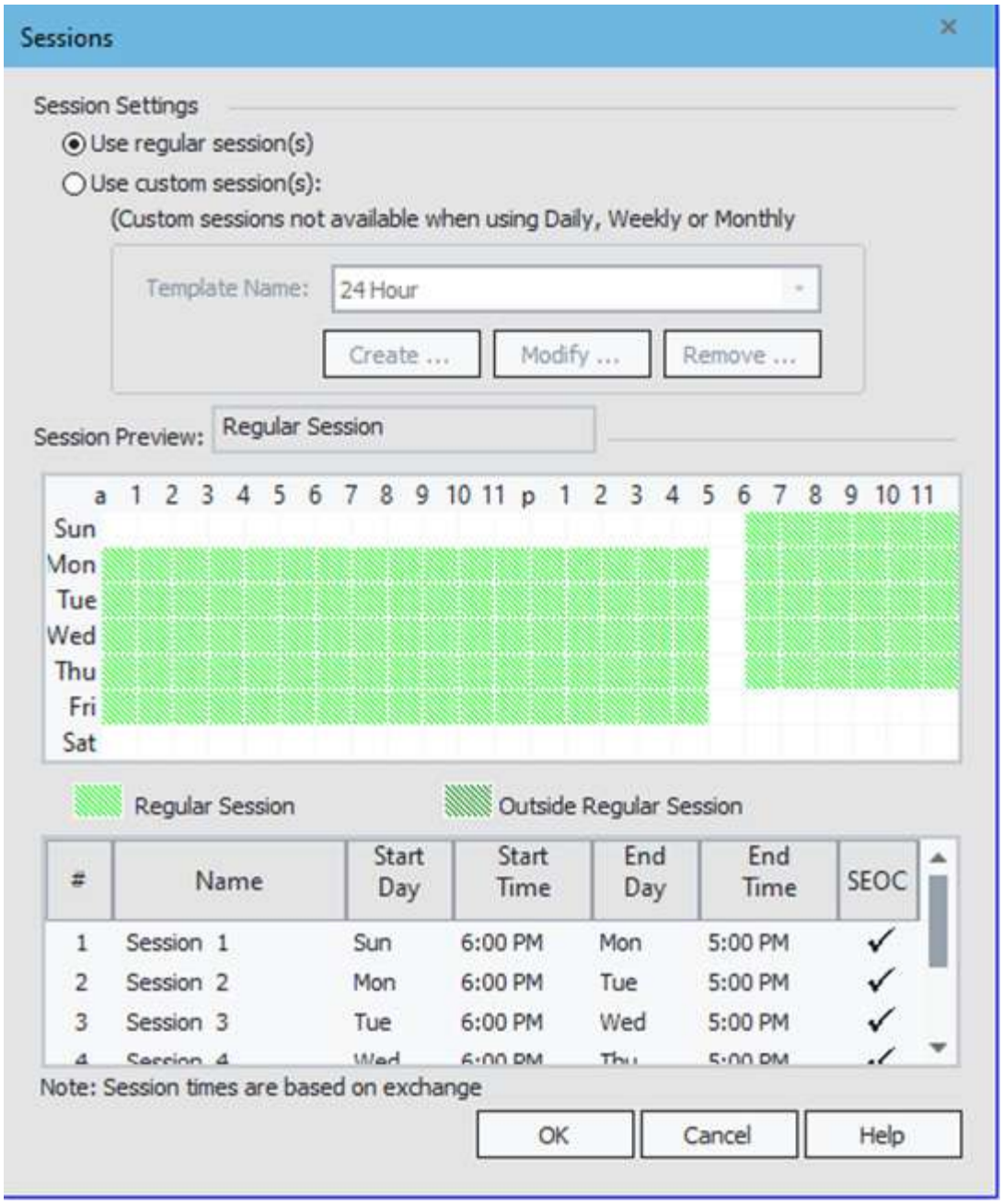


Figure 8.4 – Futures trading sessions

In *Figure 8.4*, you can see the regular session for the selected futures contract; trading sessions can be customized.

> ***NOTE***
>
> *Many traders use customize futures trading sessions because they find it useful to look at the price and volume movements only during certain trading hours.*

Now that we understand the basics of the futures market, we can start building algorithms. Among the many possible trading strategies, the **time breakout** approach is one I've found both effective and easy to understand.

How to write algorithms for time breakout strategies

Time breakout refers to a trading strategy that focuses on the timing of price movements within a certain period. This strategy aims to identify significant price changes that occur at specific times of the trading day or within certain timeframes. The rationale behind this logic is that certain times of the day consistently show higher volatility, providing predictable opportunities.

The logic behind this strategy is that the highs and lows formed at specific times could be defined as support or resistance. In this section, we are going to learn about the following:

- How to define time support/resistance levels in EasyLanguage
- How to write a time breakout strategy
- How to use sensitivity analysis to study the market

As you may recall from the earlier chapters, any time we approach a new market or a new strategy, it is advisable to start by developing indicators first.

How to define time support and resistance in EasyLanguage

We can identify time support and resistance by writing the following code:

```
var:Resistance(0),Support(0);
if time=800 then Begin
    Resistance = highest(H,24);
    Support    = lowest(L,24);
End;
plot1(Support,"Support");
plot2(Resistance,"Resistance");
```

In *Figure 8.5*, you can see what the previous indicator looks like in a chart:

Figure 8.5 – The Time Support-Resistance indicator on a gold future

In *Figure 8.5*, you can see the Time Support-Resistance indicator applied to a 60-minute gold futures chart, where the blue lines show the time resistance and the red lines show the support.

In this example, every morning at 8:00 a.m., the algorithm calculates **`Resistance`** at the highest high of the last 24 bars, and **`Support`** at the lowest low of the last 24 bars.

Starting from this indicator's script, let's see now how to write a complete time breakout strategy:

How to write the complete script

Here, you will find the script to algorithmically trade using a time breakout strategy on futures:

```
input:SR_Time(800),SR_BarsBack(12),WaitBars(1);
var:Resistance(0),Support(0),SR_count(0);
if time=SR_Time then Begin
    Resistance=highest(H,SR_BarsBack);
    Support=lowest(L,SR_BarsBack);
    SR_count=0;
End;
SR_count=SR_count+1;
//Entry
if time>=SR_Time and SR_count<=Waitbars then begin
    buy       next bar Resistance stop;
    sellshort next bar Support stop;
End;
```

As you will be able to test yourself, the performance of a breakout strategy varies significantly depending on how much time has passed since the support/resistance levels formed; this is why, in this script, I have added a new input, **`WaitBars`**, with the purpose of executing the trade only within **`WaitBars`** from **`SR_Time`**.

In *Figure 8.6*, you can see some entry signals on the 60-minute gold futures chart.

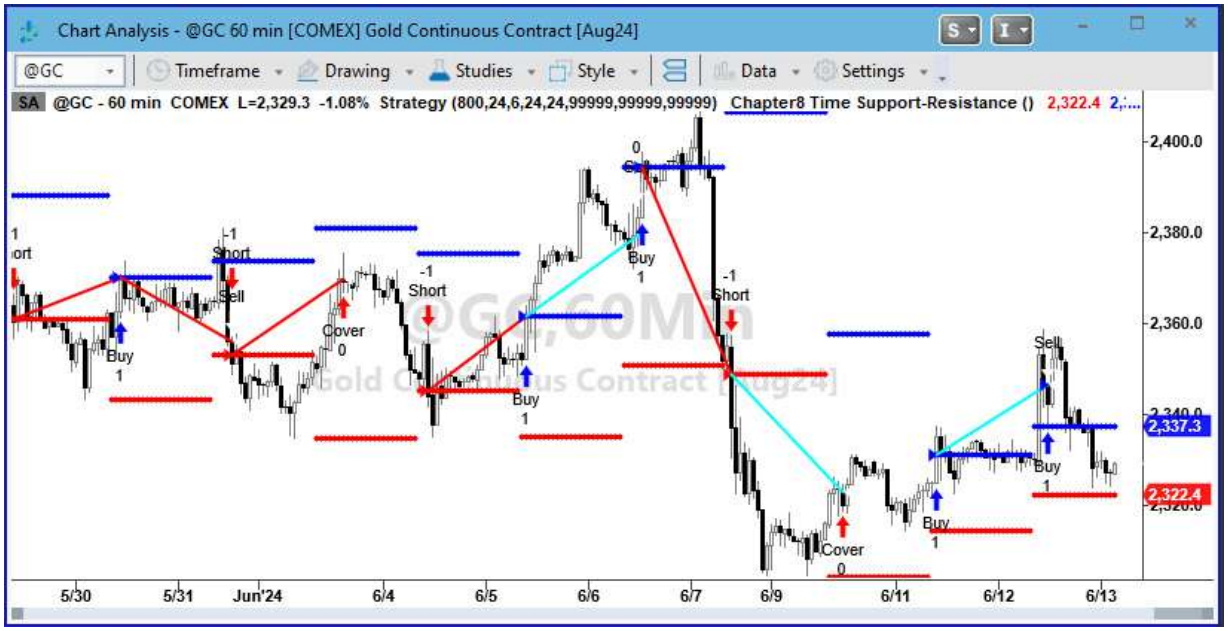


Figure 8.6 – Time BreakOut on a gold future

In *Figure 8.6*, you can see how the algorithm manages entries/exits. The strategy has no exit rules but always switches direction from long to short and vice versa each time there is a new signal. The purpose of such an approach is to understand how prices react when they break a support or resistance level over time.

How to use sensitivity analysis to study the market

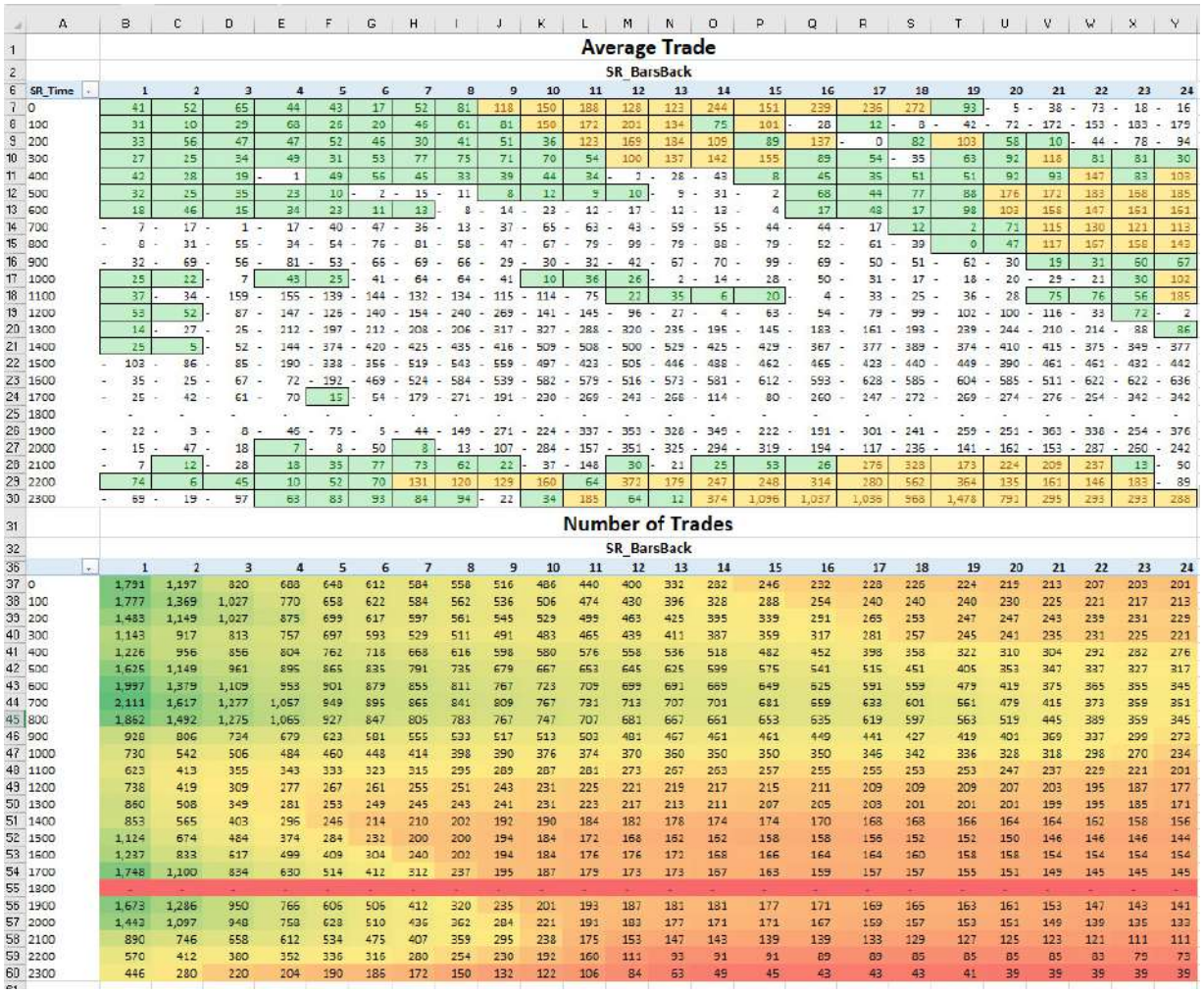
Let's run the following optimization on the 60-minute chart's gold futures continuous contract (@GC) from April 2020 to May 2024 (1,000 days), as shown in the settings in *Table 8.2*:

Input	Start	Stop	Increment
`SR_Time`	0	2,400	100
`SR_BarsBack`	1	24	1
`WaitBars`	6	6	1

Table 8.2 – The settings for sensitivity analysis on @GC time breakout

Note that, in *Table 8.2*, we only optimize `SR_Time` and `SR_BarsBack`, leaving `WaitBars` unchanged at 6. You can try to launch the same analysis with different `WaitBars`.

In *Figure 8.7*, you can see the **Average Trade** and **Number of Trades** heat maps, obtained with the Print Log method we learned in the previous chapters:



Average Trade

SR_BarsBack

SR_Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
0	41	52	65	44	43	17	52	81	118	150	188	128	123	244	151	239	236	272	93	-5	-38	-73	-18	-16
100	31	10	29	68	26	20	46	61	81	150	172	201	134	75	101	-28	12	-8	-42	-72	-172	-153	-133	-179
200	33	56	47	47	52	46	30	41	51	36	123	169	134	109	89	137	-0	82	103	58	10	-44	-78	-94
300	27	25	34	49	31	53	77	75	71	70	54	100	137	142	155	89	54	-35	63	92	118	81	81	30
400	42	28	19	-1	49	56	45	33	39	44	34	-2	28	43	8	45	35	51	51	92	93	147	83	103
500	32	25	35	23	10	-2	15	11	8	12	9	10	-9	31	2	68	44	77	88	176	172	183	158	185
600	18	46	15	34	23	11	13	-8	14	23	12	17	11	13	4	17	48	17	98	103	158	147	161	161
700	-7	-17	-1	-17	-40	-47	-36	-13	-37	-65	-63	-43	-59	-55	-44	-44	-17	12	2	71	115	130	121	113
800	-8	-31	-55	-34	-54	-76	-81	-58	-47	-67	-79	-99	-79	-88	-79	-52	-61	-39	0	47	117	167	158	143
900	-32	-69	-56	-81	-53	-66	-69	-66	-29	-30	-32	-42	-67	-70	-99	-69	-50	-51	-62	-30	19	31	60	67
1000	25	22	-7	43	25	-41	-64	-64	-41	10	36	26	-2	-14	-28	-50	-31	-17	-18	-20	-25	-21	30	102
1100	37	-34	-159	-155	-139	-144	-132	-134	-115	-114	-75	21	35	6	20	-4	-33	-25	-36	-28	75	76	56	185
1200	53	52	-87	-147	-126	-140	-154	-240	-269	-141	-145	-96	-27	-4	-63	-54	-79	-99	-102	-100	-116	-33	72	-2
1300	14	-27	-25	-212	-197	-212	-208	-206	-317	-327	-288	-320	-236	-195	-145	-183	-161	-193	-239	-244	-210	-214	-88	86
1400	25	5	-52	-144	-374	-420	-425	-435	-416	-509	-508	-500	-529	-425	-429	-367	-377	-389	-374	-410	-415	-375	-345	-377
1500	-103	-86	-85	-190	-338	-356	-519	-543	-559	-497	-423	-505	-446	-488	-462	-465	-423	-440	-449	-390	-461	-461	-432	-442
1600	-35	-25	-67	-72	-192	-469	-524	-584	-539	-582	-579	-516	-573	-581	-612	-593	-628	-585	-604	-585	-511	-622	-622	-636
1700	-25	-42	-61	-70	-15	-54	-179	-271	-191	-230	-269	-243	-268	-114	-80	-260	-247	-272	-269	-274	-276	-254	-342	-342
1800	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
1900	-22	-3	-8	-46	-75	-5	-44	-149	-271	-224	-337	-353	-328	-349	-222	-151	-301	-241	-259	-251	-363	-338	-254	-376
2000	-15	-47	-18	-7	-8	-50	-8	-13	-107	-284	-157	-351	-325	-294	-319	-194	-117	-236	-141	-162	-153	-287	-260	-242
2100	-7	12	-28	18	35	77	73	62	22	-37	-148	30	-21	25	53	26	176	328	173	224	209	237	13	-50
2200	74	6	45	10	52	70	131	120	129	160	64	372	179	247	248	314	280	562	364	135	161	146	133	-39
2300	-65	-19	-57	63	83	93	84	94	-22	34	185	64	12	374	1,096	1,037	1,096	968	1,478	791	295	293	298	288

Number of Trades

SR_BarsBack

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
0	1,791	1,197	820	688	648	612	584	558	516	486	440	400	332	282	246	232	228	226	224	219	213	207	203	201
100	1,777	1,369	1,027	770	658	622	584	562	536	506	474	430	396	328	288	254	240	240	240	230	225	221	217	213
200	1,483	1,149	1,027	875	699	617	597	561	545	529	495	463	425	395	339	251	265	253	247	247	243	239	231	229
300	1,143	917	813	757	697	593	529	511	491	483	465	439	411	387	359	317	281	257	245	241	235	231	225	221
400	1,226	956	856	804	762	718	668	616	598	580	576	558	536	518	482	452	398	358	322	310	304	292	282	276
500	1,625	1,149	961	895	865	835	791	735	679	667	653	645	625	599	575	541	515	451	405	353	347	337	327	317
600	1,997	1,379	1,109	953	901	879	855	811	767	723	709	699	691	689	649	625	591	559	479	419	375	365	355	345
700	2,111	1,617	1,277	1,057	949	895	865	841	809	767	731	713	707	701	681	659	633	601	561	479	415	373	359	351
800	1,862	1,492	1,275	1,065	927	847	805	783	767	747	707	681	667	661	653	635	619	597	563	519	445	389	359	345
900	928	806	724	679	623	591	555	533	517	513	503	481	467	461	461	449	441	427	419	401	369	337	299	273
1000	730	542	506	484	460	448	414	398	390	376	374	370	360	350	350	350	346	342	336	328	318	298	270	234
1100	623	413	355	343	333	323	315	295	289	287	281	273	267	263	257	255	255	253	253	247	237	229	221	201
1200	738	419	309	277	267	261	255	251	243	231	225	221	219	217	215	211	209	209	209	207	203	195	187	177
1300	860	508	349	281	253	249	245	243	241	231	223	217	213	211	207	205	203	201	201	201	199	195	185	171
1400	853	565	403	295	246	214	210	202	192	190	184	182	178	174	174	170	168	168	166	164	164	162	158	156
1500	1,124	674	484	374	284	232	200	200	194	184	172	168	162	162	158	158	156	152	152	150	146	146	146	144
1600	1,237	833	617	499	409	304	240	202	194	184	176	176	172	168	166	164	164	160	158	158	154	154	154	154
1700	1,748	1,100	834	630	514	412	312	237	195	187	179	173	173	167	163	159	157	157	155	151	149	145	145	145
1800	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
1900	1,673	1,286	950	766	606	506	412	320	235	201	193	187	181	181	177	171	169	165	163	161	153	147	143	141
2000	1,442	1,097	948	758	628	510	436	362	284	221	191	183	177	171	171	167	159	157	153	151	149	139	135	133
2100	890	746	558	612	534	475	407	359	295	238	175	153	147	143	139	139	133	129	127	125	123	121	111	111
2200	570	412	360	352	336	316	280	254	230	192	160	111	93	91	91	89	89	85	85	85	85	83	75	73
2300	446	280	220	204	190	186	172	150	132	122	106	84	63	49	45	43	43	43	41	39	39	39	39	39

Figure 8.7 – The Average Trade and Number of Trade gold futures heat maps

By looking at *Figure 8.7*, we can deduce the following:

- Early breakouts from midnight to 6 a.m. are profitable
- Breakouts that occurred from 7 a.m. to 8 p.m. are not profitable
- Breakouts that occurred from 9 p.m. to 11 p.m. are profitable

In *Table 8.3*, you can see the parameters where the average trade profit is higher than $100:

#	**SR_Time**	**SR_BarsBack**
1	0–300	12–13

#	SR_Time	SR_BarsBack
2	500–800	21–24
3	2,100–2,300	17–22

Table 8.3 – Parameters showing an Average Trade Profit higher than $100

In *Table 8.3*, you can see the settings pairs (`SR_Time` and `SR_BarsBack`) showing an average trade higher than $100.

The average trade profit is essential for evaluating a trading strategy, as it must cover transaction costs such as brokerage fees and slippage. This profit needs to be high enough to ensure net profitability. In futures trading, it's important to account for the changing value of futures contracts when calculating the average trade profit.

Let's delve into this important concept. When trading a futures contract, the unit of measurement is one contract, not a dollar amount like in stock trading or, as we will see, forex. Therefore, when you buy a futures contract, the value of your purchase changes depending on the price.

In *Figure 8.8*, you can see the gold futures contract value for the last 20 years:

Figure 8.8 – The gold futures contract value and trade size

In *Figure 8.8*, you can see the trend of the value of a gold futures contract over the last 20 years. You can see that the value of one contract varies significantly over time and has almost doubled since 2004. This means that when conducting long-term backtesting, it is advisable to set strategies to trade a fixed USD capital with a variable number of contracts, rather than a fixed number of contracts. In the second sub-chart of the same figure, you can see that to trade 1 million USD, you would have

needed 10 contracts at the beginning of 2004 and fewer than 5 contracts in 2024. Such an approach will allow you to better evaluate the average net profit calculated by TradeStation's Performance Summary.

In this section, we have seen that the futures market is more complex than the stock market, which we discussed in previous chapters, as it requires algorithmic traders to acquire specific knowledge about the basic mechanisms of a contract that are not directly related to programming. Next, we will explore the forex market, which offers a distinct set of concepts compared to what we have covered thus far.

Forex algorithmic trading

Forex, or **foreign exchange**, refers to the global marketplace for buying and selling national currencies. It is the largest and most liquid financial market in the world. Forex trading is primarily conducted **over the counter** (**OTC**), meaning that it takes place directly between two parties without the supervision of an exchange. This setup distinguishes it from exchange-traded markets, such as stock exchanges and futures exchanges, and contributes to the unique nature of the forex market.

In this section, we are going to learn the following:

- The basic concepts of forex
- How to write algorithms for the forex market

The basic concepts of forex

As with the futures market, there are basic concepts you need to know about forex trading. In this section, we will discuss the following:

- Currency pairs
- Interest rates
- Economic indicators
- Rollover
- Margin trading
- Forex trading sessions
- Major versus minor forex crosses

Currency pairs

Forex trading involves currency pairs such as EUR/USD (euro/US dollar) or GBP/JPY (British pound/Japanese yen). In each pair, the first currency is known as the **base currency**, and the second

currency is referred to as the **quote currency**. The value of the base currency is measured in terms of the quote currency. The price of the pair indicates how much of the quote currency is needed to purchase one unit of the base currency. So, for example, if EURUSD quotes 1.0740, it means that 1 euro (EUR) is equivalent to 1.0740 **US dollars** (**USD**); in other words, you would need 1.0740 USD to buy 1 EUR.

Interest rates

Central banks' interest rate decisions are one of the most critical factors in the forex market. Higher interest rates typically attract foreign capital and can strengthen a currency, while lower rates can have the opposite effect.

A forex trader cannot function without information on interest rates and central banks' meeting dates.

In *Figure 8.9*, you can see the central banks' rates and the next meeting as of June 12, 2024.

Central Banks Rates »

Central Banks	Interest Rates	Next Meeting
FED	5.50%	Jun 12, 2024
ECB	4.25%	Jul 18, 2024
BOE	5.25%	Jun 20, 2024
SNB	1.50%	Jun 20, 2024
RBA	4.35%	Jun 18, 2024
BOC	4.75%	Jul 24, 2024
RBNZ	5.50%	Jul 10, 2024
BOJ	0.10%	Jun 14, 2024
CBR	16.00%	Jul 26, 2024
RBI	6.50%	Aug 08, 2024
PBOC	3.45%	
BCB	10.50%	Jun 19, 2024

Figure 8.9 – The central banks' rates and the next meeting dates

In *Figure 8.9*, you can see the central banks' rates and the next meeting date on June 12, 2024. You can find more up-to-date information at https://www.investing.com/economic-calendar/.

By looking at the central banks' interest rates, we can see significant differences among different countries. This disparity is the main driver for increased volatility.

Economic indicators

Each currency pair represents the economic power dynamics between two nations. Therefore, any event that strengthens one nation's economy relative to another also strengthens its currency exchange rate. The most relevant key economic power indicators are as follows:

- **Gross domestic product (GDP)**:
- Employment data, especially **non-farm payrolls** (**NFPs**) in the US
- Inflation rates – **consumer price index** (**CPI**) and **producer price index** (**PPI**)
- Retail sales
- Industrial production
- Trade balance

Keep in mind that such economic indicators have to be analyzed in pairs. This is the only way they can provide insights into the economic health of a country and move the forex cross significantly: for example, consider the EUR/USD currency pair; if the European GDP improves by, say, 3%, this alone may not increase the EUR/USD price unless the US GDP either declines or improves at a slower rate.

Note that, in TradeStation, the forex symbols do not have any characters between the first and second currency. For example, EUR versus USD becomes EURUSD, USD versus Japanese yen becomes USDJPY, and so on.

Forex rollover

Forex rollover, also known as **swap** or **overnight interest**, refers to the interest paid or earned to hold a currency position overnight in the forex market. The forex market operates 24 hours a day, but there is a specific time (typically, 5 PM EST) when trades are considered to have rolled over to the next day. At this point, positions may incur a rollover fee or earn interest, depending on the interest rate differential between the two currencies involved in the pair.

Futures rollovers and forex rollovers have different implications; **futures rollover** involves the process of closing out a position in a futures contract that is nearing expiration and, simultaneously, opening a position in a contract with a later expiration date. Conversely, **forex rollover** refers to the interest paid or earned to hold a forex position overnight. In other words, forex rollover doesn't imply any expiration. When you trade currency crosses with significantly different interest rates, such as USDJPY, forex rollover must be closely monitored. For instance, trading a short strategy on USDJPY could be more costly, as your strategies must yield profits greater than the interest rate differential (5.50% - 0.10% = 5.40%) to break even.

Margin trading

Forex margin trading is a method that allows traders to buy or sell currency pairs with a fraction of the full value of the position, known as the **margin**. It is essentially a loan provided by the broker to the trader, enabling them to leverage their trades and control larger positions than their actual account balance would allow.

For example, suppose you have a trading account with $1,000 and your broker offers 100:1 leverage. You want to open a position in the EUR/USD pair worth $100,000. With 100:1 leverage, the margin required to open this position is $1,000 (1% of $100,000), summarized as follows:

- **Position size**: $100,000
- **Leverage**: 100:1
- **Margin required**: $1,000

If the EUR/USD exchange rate moves favorably by 1%, your profit would be $1,000 (1% of $100,000), doubling your initial investment. However, if the market moves unfavorably by 1%, you would lose your entire $1,000 margin. Therefore, you need to use leverage according to your risk tolerance

Forex trading sessions

The forex market operates 24 hours a day, 5 days a week, due to the global nature of currency trading. The market is divided into four major trading sessions – Sydney, Tokyo, London, and New York. Understanding the forex market hours is crucial for traders to optimize their trading strategies, as different sessions have varying levels of volatility and liquidity. *Table 8.4* gives you a detailed breakdown of the major forex trading sessions:

Session	**Hours**
Sydney session	**Hours (EST)**: 5:00 p.m. to 2:00 a.m. EST
Tokyo session	**Hours (EST)**: 7:00 p.m. to 4:00 a.m. EST
London session	**Hours (GMT)**: 8:00 a.m. to 5:00 p.m. GMT
New York session	**Hours (GMT)**: 1:00 p.m. to 10:00 p.m. GMT

Table 8.4 – The major trading sessions

From *Table 8.4*, you can see the four major forex trading sessions:

- The **Sydney session** is where the trading day officially begins. It’s typically less volatile compared to the other sessions but can see increased activity when significant economic news is released from Australia and New Zealand.

- The **Tokyo session** overlaps with the Sydney session for a few hours, which can result in increased trading volume. This session is known for movements in the yen pairs and significant influence from economic news from Japan.
- The **London session** is the most active and liquid session, due to the large number of forex traders and financial institutions in the market. It overlaps with both the Tokyo session at its start and the New York session at its end, leading to higher volatility and trading volume.
- The **New York session** is the second most active session and is particularly important due to the presence of major US economic data releases. The overlap with the London session creates the most volatile period of the trading day.

Understanding the forex market hours and the characteristics of each trading session can help traders optimize their trading strategies by selecting the best times to trade, based on their goals and the currency pairs they are interested in. The most significant trading opportunities typically arise during the overlaps between major sessions, particularly between London and New York, due to the high liquidity and volatility.

In *Figure 8.10*, you can see the forex **Sessions** window from TradeStation charts:

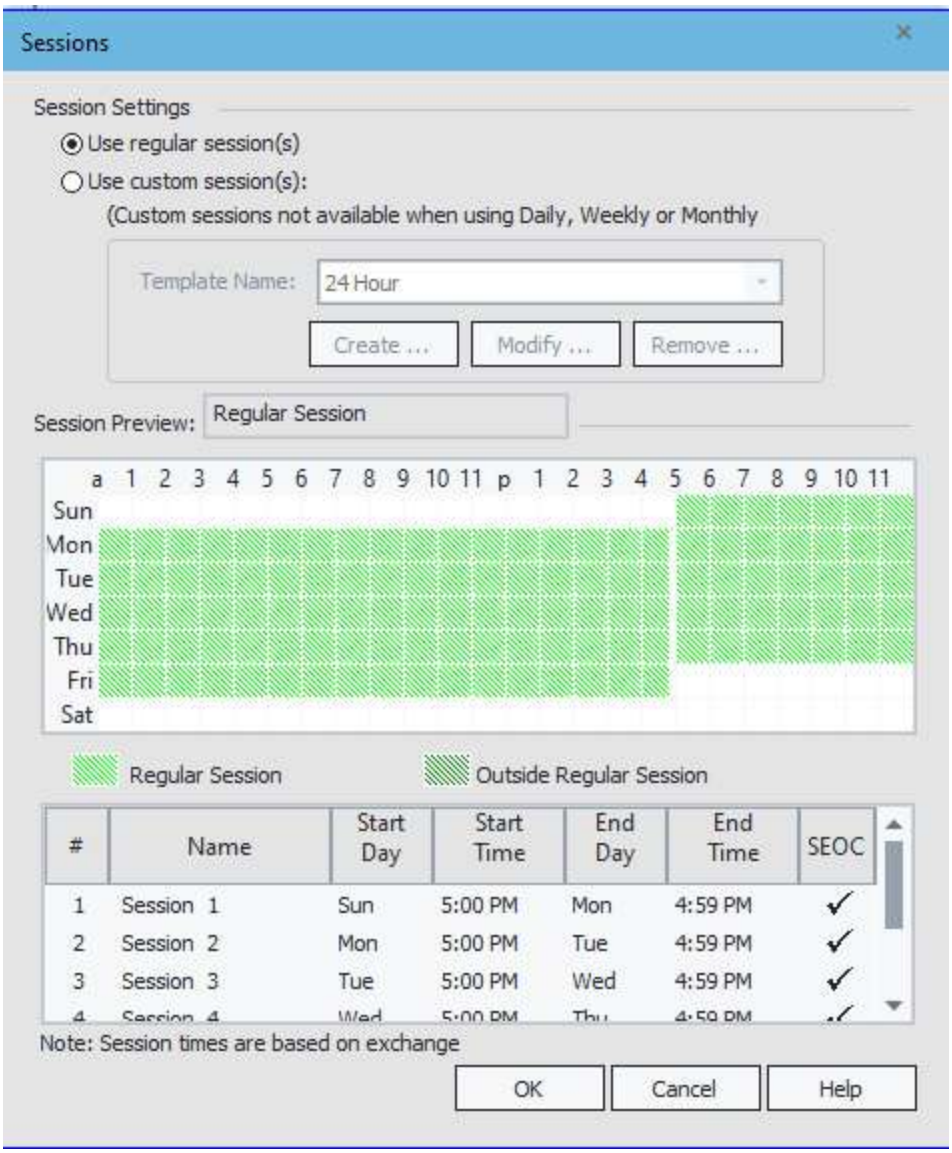


Figure 8.10 – The TradeStation forex Sessions window

In *Figure 8.10*, you can see the TradeStation forex Sessions window, which you can open from the main chart menu by going to **Data** | **Edit Symbol** | **Properties**.

Major versus minor forex crosses

In the forex market, currency pairs are commonly categorized as **major pairs** and **minor pairs**, based on their liquidity, trading volume, and popularity among traders. This classification helps traders identify the most actively traded pairs and those with potentially higher volatility. When discussing forex crosses, the terms “major crosses” and “minor crosses” are also used but refer specifically to currency pairs that exclude the US dollar (USD):

- **Major crosses** involve currency pairs that exclude the USD but consist of two major currencies traded against each other. For example, EURGBP represents the exchange rate between the EUR and GBP.
- **Minor crosses** involve currency pairs that exclude the US dollar (USD) but consist of one major currency and one currency from a smaller or emerging market economy. For example, EURMXN represents the exchange rate between the Euro (EUR) and the Mexican peso (MXN).

Minor crosses may be more susceptible to economic and geopolitical developments specific to the countries involved in the pair. While they may offer trading opportunities for experienced traders, they are often subject to lower trading volumes and may exhibit greater price fluctuations during periods of market uncertainty.

Now that we know the basics of the forex market, we can start building some trading algorithms.

How to write algorithms for the forex market

Many opportunities for algorithmic and quantitative trading in forex are generated by the overlap of trading sessions throughout the day.

In this section, we are going to do the following:

- Analyze trading session overlaps and intraday forex volatility
- Delve into the concept of breakout versus fake breakout
- Write forex breakout algorithms
- Write fake breakout algorithms

Analyzing trading session overlaps and intraday forex volatility

In *Figure 8.11*, you can see how the four main trading sessions overlap during the day.

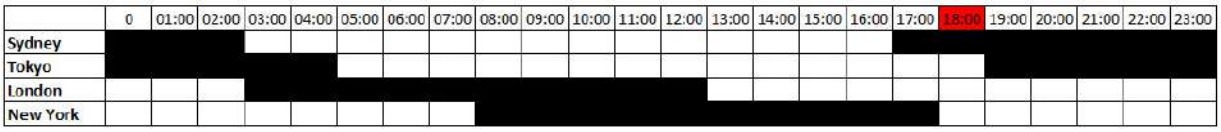


Figure 8.11 – Forex session overlaps

By looking at *Figure 8.11*, let's try to analyze the intraday forex volatility. To achieve such a task, let's apply the following script to 60-minute forex charts:

```
var:RangePct(0);
RangePct= High / Low -1;
if barstatus(1)=2 then print(symbol,",",date,",",time,",",RangePct*100);
```

When the preceding code is applied to 60-minute forex symbols, it will print the hourly range extension as a percentage in the TradeStation Print Log. From there, you can copy and paste the data into an Excel spreadsheet, following the same procedure we learned in the previous chapters.

In *Figure 8.12*, you can see the average intraday volatility for four forex symbols:

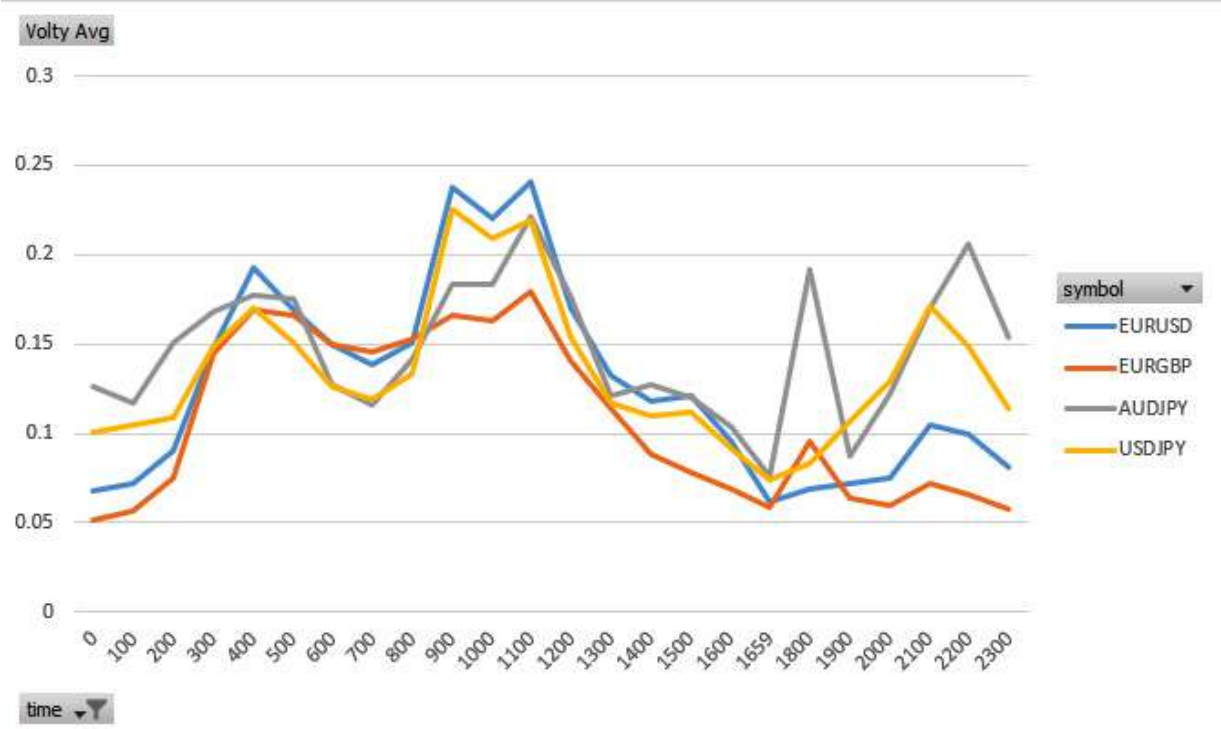


Figure 8.12 – Forex intraday volatility

In *Figure 8.12*, you can see the average intraday volatility for four different forex crosses. As you can see, all of them show the highest volatility peaks during the start of the London session (3 a.m.) and the start of the New York session (8 a.m.). AUDJPY and USDJPY show also a volatility peak after 5 p.m. (which is after Sydney and Tokyo open).

But how can we use such analysis and information to create an algorithmic strategy?

The question that forex traders most frequently ask whenever volatility increases is, are we facing a breakout or a fake breakout? Let's try to answer this question with the help of some algorithms.

Breakout versus fake breakout

Distinguishing between a genuine breakout and a fake breakout (also known as a **false breakout**) is a significant challenge for traders, particularly in the highly liquid and often volatile forex market. Here's a detailed look at both concepts and strategies to help traders identify and manage them effectively.

What is a breakout?

A **breakout** occurs when the price of a currency pair moves above a resistance level or below a support level with increased volume. This is typically seen as a signal that the price will continue in the direction of the breakout, potentially starting a new trend.

What is a fake breakout?

A **fake breakout** happens when the price moves beyond a support or resistance level but then quickly reverses direction, trapping traders who entered positions based on the initial breakout. This can lead to losses for traders who anticipated a continuation in the breakout direction.

Let's analyze a way to algorithmically manage this crucial aspect by applying the same code we wrote for the futures breakout to USDJPY, and then run a sensitivity analysis on the 60-minute chart from April 2020 to May 2024 (1,000 days), according to the same sensitivity analysis settings we used in *Table 8.2* at the beginning of this chapter.

In *Figure 8.13*, you can see the **Average Trade** profit Excel heat map:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
1										**Average Trade**															
2										SR_BarsBack															
6	SR_Tir	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
7	0	2,143	3,142	3,757	3,828	4,141	4,923	4,867	4,554	4,274	3,945	4,227	4,134	3,571	2,781	2,655	3,564	3,526	3,342	3,506	3,307	3,091	3,772	3,376	3,799
8	100	724	1,445	2,172	2,208	3,275	3,104	3,690	3,594	3,257	3,536	3,032	3,127	3,197	3,002	2,555	1,927	2,637	2,782	2,328	2,223	2,396	2,566	3,336	3,030
9	200	703	347	1,262	1,728	1,753	2,590	2,263	2,208	2,031	1,708	2,008	1,506	1,587	1,907	1,983	1,254	1,099	1,612	1,560	1,472	1,551	1,728	1,861	2,800
10	300	344	178	290	930	861	811	1,430	1,148	938	1,026	1,051	1,402	1,279	1,288	1,734	1,606	1,212	779	1,455	1,657	1,833	1,686	1,774	1,740
11	400	3,560	3,246	3,598	3,828	4,460	4,154	4,561	4,904	4,454	3,959	3,957	3,962	4,041	3,941	3,823	3,759	3,285	3,730	3,617	3,804	3,853	4,112	4,269	4,612
12	500	2,263	3,060	3,419	3,074	3,710	3,707	3,619	3,675	4,366	4,635	4,261	4,441	4,414	4,490	4,338	4,095	4,241	4,053	4,405	3,928	3,395	3,760	3,569	3,832
13	600	1,894	2,390	2,476	2,831	3,005	3,164	3,294	3,525	3,700	3,987	4,259	4,262	4,405	4,309	4,182	4,297	4,024	4,127	3,853	4,494	3,495	3,663	3,669	3,531
14	700	2,790	2,899	3,383	3,511	3,880	3,976	4,131	4,336	4,733	4,727	4,920	5,179	4,918	5,284	5,285	5,612	5,726	5,439	5,411	5,021	5,548	4,798	4,757	4,836
15	800	2,605	3,294	2,957	3,754	3,989	4,111	4,006	4,058	4,128	4,481	4,560	5,005	5,192	4,853	5,212	5,269	5,291	5,520	5,322	5,188	4,780	5,022	4,744	4,099
16	900	1,670	1,607	1,506	1,892	2,682	2,954	3,076	3,226	3,231	3,178	3,761	3,639	4,189	3,931	3,742	4,129	4,331	4,360	4,297	4,240	4,194	5,522	3,484	3,318
17	1000	2,118	2,485	1,906	1,660	2,113	2,966	3,588	3,421	4,056	4,270	4,169	4,806	4,765	5,469	5,108	4,973	5,404	5,442	5,571	5,493	5,391	5,523	5,701	5,512
18	1100	1,617	2,306	2,233	1,916	2,280	2,296	2,571	2,761	2,897	2,622	2,784	2,810	3,478	3,429	4,044	3,768	3,700	4,042	3,992	4,074	4,293	4,364	4,385	4,151
19	1200	483	1,728	2,280	2,830	2,235	2,076	2,449	2,662	3,127	3,369	3,740	4,047	4,168	4,862	4,788	5,447	4,982	4,905	5,280	5,219	5,336	5,348	5,439	5,626
20	1300	919	725	1,752	1,888	3,203	2,507	2,586	3,017	3,376	3,568	3,468	3,838	4,421	4,466	5,213	4,991	5,463	4,962	4,917	4,925	4,817	4,978	4,929	4,931
21	1400	355	760	685	1,164	1,604	2,312	1,746	2,350	2,455	2,710	2,830	3,414	3,811	4,114	4,180	4,506	4,360	4,856	4,586	4,536	4,502	4,420	4,461	4,448
22	1500	- 660	- 732	- 342	692	2,162	3,197	4,166	3,642	3,930	3,910	4,190	4,598	5,073	4,998	5,118	4,979	5,321	4,766	5,019	4,619	4,491	4,683	4,601	4,625
23	1600	- 138	- 180	253	839	1,172	2,140	3,789	4,406	4,250	4,365	4,498	4,330	5,290	5,551	5,498	5,594	5,710	6,108	6,003	6,398	5,904	6,019	6,074	5,972
24	1800	1,055	1,352	1,258	1,557	1,770	1,362	977	1,753	2,946	2,933	2,231	2,485	2,768	2,424	2,523	3,225	3,045	2,971	3,763	4,357	4,044	4,405	4,156	4,097
25	1900	542	839	660	412	- 27	321	287	- 85	754	2,000	2,453	2,073	2,038	2,382	2,051	2,163	2,395	2,222	2,145	2,775	3,346	3,145	3,661	3,544
26	2000	1,038	888	1,012	795	369	132	457	265	- 241	941	2,001	2,909	2,689	2,449	2,894	2,417	2,621	2,999	2,799	2,775	3,285	3,985	3,963	4,049
27	2100	2,760	1,690	1,673	1,479	1,074	916	790	1,356	1,099	- 117	1,132	2,415	3,146	2,884	2,595	3,214	2,848	3,005	3,117	3,155	3,127	3,876	4,252	4,347
28	2200	1,293	2,238	587	767	364	- 73	- 241	- 800	- 429	- 263	- 677	1,195	1,213	1,963	2,207	1,892	2,431	2,545	3,049	3,202	3,277	3,495	3,891	4,129
29	2300	2,118	3,638	2,598	1,290	1,487	799	694	612	- 96	- 271	783	- 74	1,284	1,010	2,265	2,773	2,327	2,649	3,219	3,721	3,013	3,109	3,641	3,641

Figure 8.13 – A USJPY breakout versus a fake breakout – the Average Trade profit Excel heat map

In *Figure 8.13*, you can see a sensitivity analysis conducted on USDJPY. I highlighted cells with an average trade profit higher than 5,000 yen in yellow and cells with a negative average trade profit in red; cells with a positive average trade profit but lower than 5,000 yen are shown in white. Consider this table from both a breakout and a fake breakout perspective – for instance, in all parameter combinations where the strategy shows a positive performance, a breakout strategy would yield positive results, whereas a fake breakout strategy would have a negative performance. In the very few parameter combinations highlighted in red, the breakout strategy has a negative performance, while the fake breakout strategy shows a positive performance.

From this, we can deduce that, for USDJPY, when the price takes a well-defined direction, it is advisable to follow the breakouts.

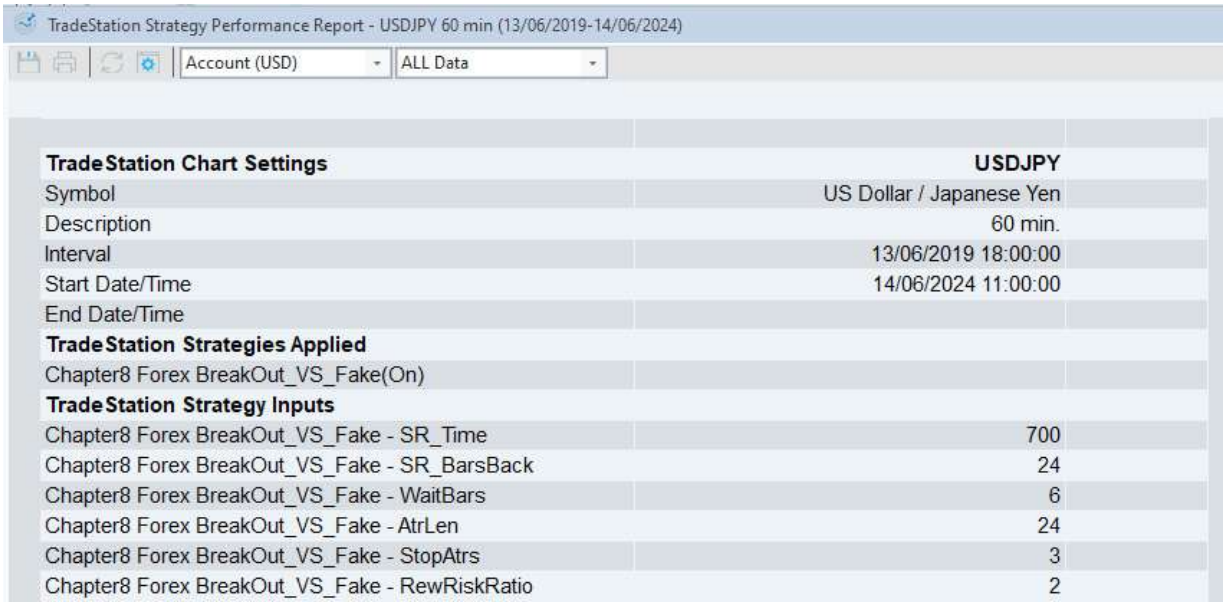
Writing forex breakout algorithms

Based on the analysis conducted so far, let's try to finalize the breakout approach on USDJPY by adding some money management components to the main block code written previously:

```
input:AtrLen(120),StopAtrs(2),RewRiskRatio(3);
var:ATR(0),TgtAtrs(0);;
ATR=Avgtruerange(AtrLen);
TgtAtrs=StopAtrs*RewRiskRatio;
if StopAtrs>0 then Begin
    if mp = 1 then sell       ("slL")next bar entryprice -
    StopAtrs*ATR stop;
    if mp =-1 then buytocover ("slS")next bar entryprice +
    StopAtrs*ATR stop;
End;
if TgtAtrs>0 then Begin
    if mp = 1 then sell       ("tgtL") next bar entryprice +
    TgtAtrs*ATR limit;
    if mp =-1 then buytocover ("tgtS") next bar entryprice -
    TgtAtrs*ATR limit;
End;
```

As you can see, this code block introduces a money management technique based on volatility stop and target, as discussed in *Chapter 7*.

Let's backtest this breakout strategy on the USDJPY 60-minute chart in the period from June 13, 2019, to 14 June, 2024, with the settings shown in *Figure 8.14*:



TradeStation Chart Settings	**USDJPY**
Symbol	US Dollar / Japanese Yen
Description	60 min.
Interval	13/06/2019 18:00:00
Start Date/Time	14/06/2024 11:00:00
End Date/Time	
TradeStation Strategies Applied	
Chapter8 Forex BreakOut_VS_Fake(On)	
TradeStation Strategy Inputs	
Chapter8 Forex BreakOut_VS_Fake - SR_Time	700
Chapter8 Forex BreakOut_VS_Fake - SR_BarsBack	24
Chapter8 Forex BreakOut_VS_Fake - WaitBars	6
Chapter8 Forex BreakOut_VS_Fake - AtrLen	24
Chapter8 Forex BreakOut_VS_Fake - StopAtrs	3
Chapter8 Forex BreakOut_VS_Fake - RewRiskRatio	2

Figure 8.14 – The breakout strategy USDJPY settings

As you can see in *Figure 8.14*, we added a money management technique with a volatility stop and target (see *Chapter 7* for more details). You can visualize some entry/exit examples in *Figure 8.15*.

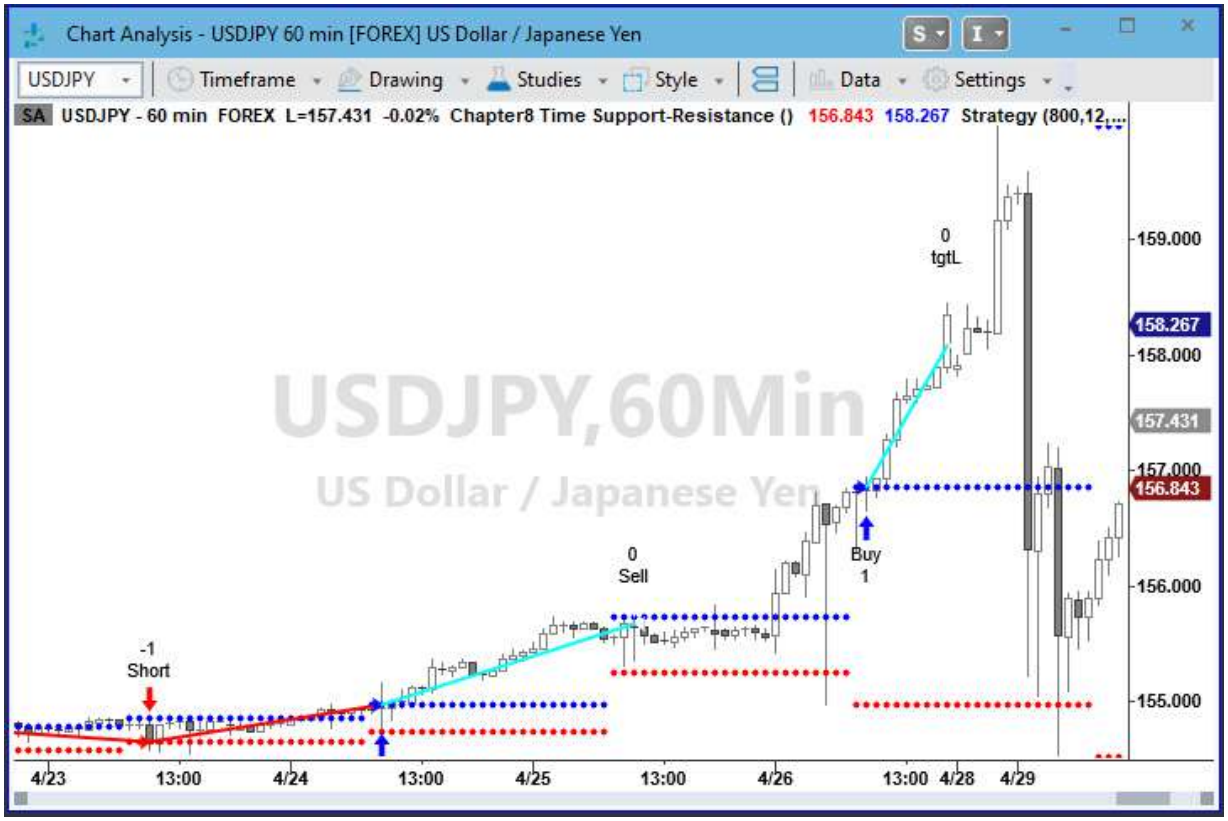


Figure 8.15 – A breakout strategy for USDJPY – entry/exit signals

As you can see in *Figure 8.15*, the strategy enters at the breakout of the support and resistance levels and applies a money management technique with stop loss and target. In *Figure 8.16*, you can see **Performance Summary**:

TradeStation Performance Summary			
All Trades			
Total Net Profit	$40,189.66	Profit Factor	1.50
Gross Profit	$120,584.98	Gross Loss	($80,395.32)
Total Number of Trades	654	Percent Profitable	41.13%
Winning Trades	269	Losing Trades	385
Even Trades	0		
Avg. Trade Net Profit	$61.45	Ratio Avg. Win:Avg. Loss	2.15
Avg. Winning Trade	$448.27	Avg. Losing Trade	($208.82)
Largest Winning Trade	$1,790.52	Largest Losing Trade	($1,147.42)
Max. Consecutive Winning Trades	8	Max. Consecutive Losing Trades	9
Avg. Bars in Winning Trades	38.64	Avg. Bars in Losing Trades	26.60
Avg. Bars in Total Trades	31.55		
Max. Shares/Contracts Held	65,000	Account Size Required	$3,347.45
Return on Initial Capital	40.19%	Annual Rate of Return	6.78%
Return Retracement Ratio	24.73	RINA Index	316.51
Trading Period	4 Years, 11 ...	Percent of Time in the Market	65.96%
Max. Equity Run-up	$40,643.95		
Max.Drawdown(Intra-Day Peak to V...		Max.Drawdown(Trade Close to Tra...	
Value	($3,834.91)	Value	($3,347.45)
Net Profit as % of Drawdown	1,047.99%	Net Profit as % of Drawdown	1,200.60%
Max. Trade Drawdown	($1,147.42)		

Figure 8.16 – The USDJPY breakout strategy Performance Summary

The **Performance Summary** table in *Figure 8.16* shows 654 trades in 4 years and 11 months, with an average trade profit of $ 61.45 (which is not bad) and a **return on account** (**ROA**) of 1,047.99% (which is a good result). You can see the equity curve of this strategy in *Figure 8.17*:

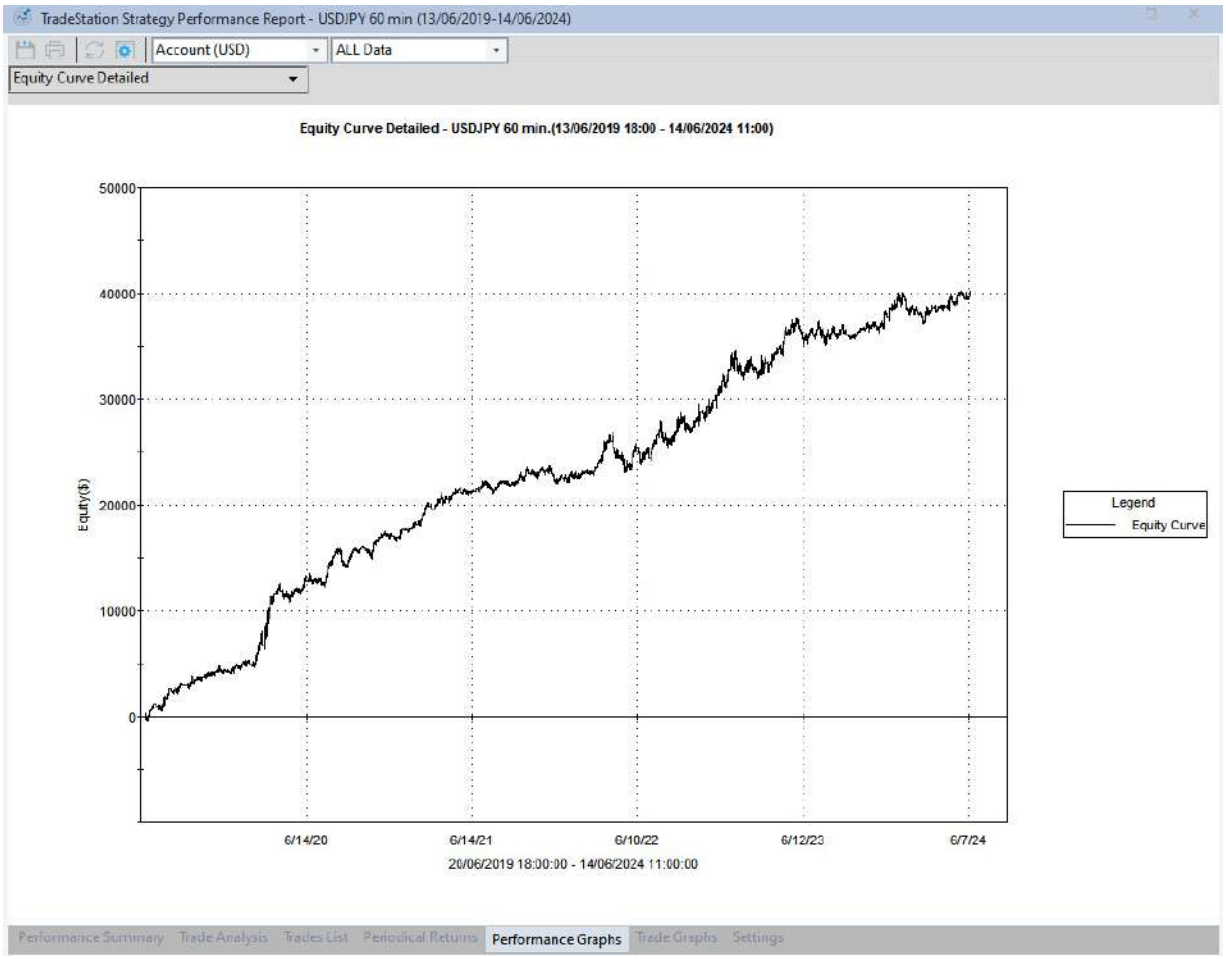


Figure 8.17 – The USDJPY breakout strategy Equity Curve

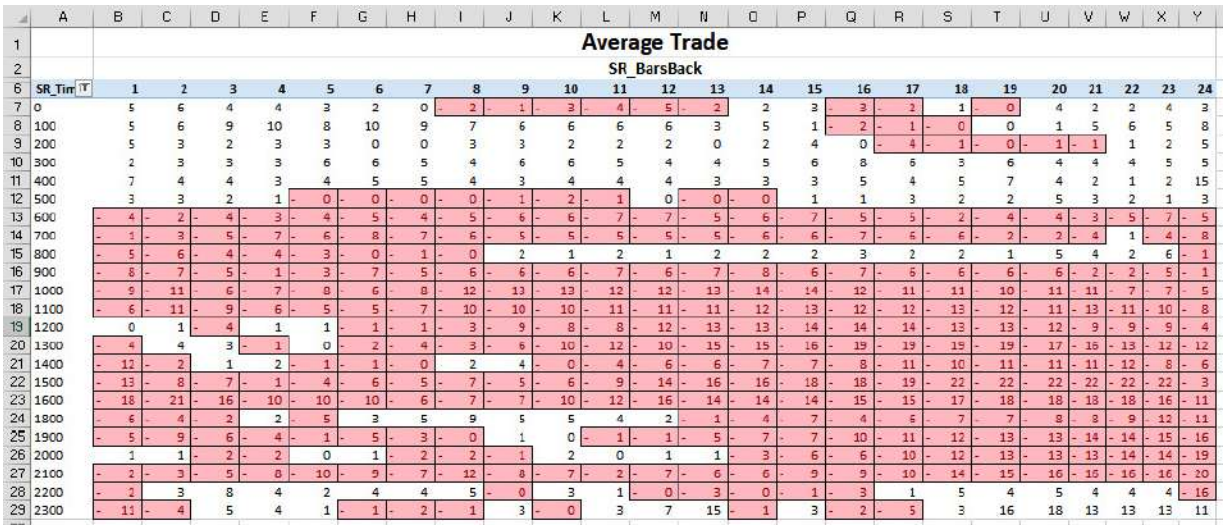
In *Figure 8.17*, you can see the equity line of the breakout strategy.

So, it seems that a breakout strategy performs well on USD JPY, but what about the flip side of the coin – that is, the fake breakout approach? From *Figure 8.13*, we know that this approach is not successful with USDJPY, so we can save time by avoiding backtesting a fake breakout on USDJPY and moving on to a different currency cross.

Writing forex fake breakout algorithms

To evaluate a fake breakout strategy, we need a less directional currency cross; let's try EURGBP. EURGBP represents the exchange rate between the **European currency** (**EUR**) and the **British currency** (**GBP**). Europe and Great Britain, in terms of interest rates and trading sessions, are much closer than the US dollar and the Japanese yen, which should give the cross a less directional trend. Let's try to run the same sensitivity analysis on EURGBP that we conducted on USDJPY.

In *Figure 8.18*, you can see the sensitivity analysis results on EURGBP:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
1		Average Trade																							
2		SR_BarsBack																							
6	SR_Tim	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
7	0	5	6	4	4	3	2	0	-2	-1	-3	-4	-5	-2	2	3	-3	-2	1	-0	4	2	2	4	3
8	100	5	6	9	10	8	10	9	7	6	6	6	6	3	5	1	-2	-1	-0	0	1	5	6	5	8
9	200	5	3	2	3	3	0	0	3	3	2	2	2	0	2	4	0	-4	-1	-0	-1	-1	1	2	5
10	300	2	3	3	3	6	6	5	4	6	6	5	4	4	5	6	8	6	3	6	4	4	4	5	5
11	400	7	4	4	3	4	5	5	4	3	4	4	4	3	3	3	5	4	5	7	4	2	1	2	15
12	500	3	3	2	1	-0	-0	-0	-0	-1	-2	-1	0	-0	-0	1	1	3	2	2	5	3	2	1	3
13	600	-4	-2	-4	-3	-4	-5	-4	-5	-6	-6	-7	-7	-5	-6	-7	-5	-5	-2	-4	-4	-3	-5	-7	-5
14	700	-1	-3	-5	-7	-6	-8	-7	-6	-5	-5	-5	-5	-5	-6	-6	-7	-6	-6	-2	-2	-4	1	-4	-8
15	800	-5	-6	-4	-4	-3	-0	-1	-0	2	1	2	1	2	2	2	3	2	2	1	5	4	2	6	-1
16	900	-8	-7	-5	-1	-3	-7	-5	-6	-6	-6	-7	-6	-7	-8	-6	-7	-5	-6	-6	-6	-2	-2	-5	-1
17	1000	-9	-11	-6	-7	-8	-6	-8	-12	-13	-13	-12	-12	-13	-14	-14	-12	-11	-11	-10	-11	-11	-7	-7	-5
18	1100	-6	-11	-9	-6	-5	-5	-7	-10	-10	-10	-11	-11	-11	-12	-13	-12	-12	-13	-12	-11	-13	-11	-10	-8
19	1200	0	1	-4	1	1	-1	-1	-3	-9	-8	-8	-12	-13	-13	-14	-14	-14	-13	-13	-12	-9	-9	-9	-4
20	1300	-4	4	3	-1	0	-2	-4	-3	-6	-10	-12	-10	-15	-15	-16	-19	-19	-19	-19	-17	-16	-13	-12	-12
21	1400	-12	-2	1	2	-1	-1	-0	2	4	-0	-4	-6	-6	-7	-7	-8	-11	-10	-11	-11	-11	-12	-8	-6
22	1500	-13	-8	-7	-1	-4	-6	-5	-7	-5	-6	-9	-14	-16	-16	-18	-18	-19	-22	-22	-22	-22	-22	-22	-3
23	1600	-18	-21	-16	-10	-10	-10	-6	-7	-7	-10	-12	-16	-14	-14	-14	-15	-15	-17	-18	-18	-18	-18	-16	-11
24	1800	-6	-4	-2	2	-5	3	5	9	5	5	4	2	-1	-4	-7	-4	-6	-7	-7	-8	-8	-9	-12	-11
25	1900	-5	-9	-6	-4	-1	-5	-3	-0	1	0	-1	-1	-5	-7	-7	-10	-11	-12	-13	-13	-14	-14	-15	-16
26	2000	1	1	-2	-2	0	1	-2	-2	-1	2	0	1	1	-3	-6	-6	-10	-12	-13	-13	-13	-14	-14	-19
27	2100	-2	-3	-5	-8	-10	-9	-7	-12	-8	-7	-2	-7	-6	-6	-9	-9	-10	-14	-15	-16	-16	-16	-16	-20
28	2200	-2	3	8	4	2	4	4	5	-0	3	1	-0	-3	-0	-1	-3	1	5	4	5	4	4	4	-16
29	2300	-11	-4	5	4	1	-1	-2	-1	3	-0	3	7	15	-1	3	-2	-5	3	16	18	13	13	13	11

Figure 8.18 – The EURGBP average trade profit heat map

In *Figure 8.18*, you can see the results of a sensitivity analysis conducted on EURGBP where I highlighted in red the negative average trade profit, while the positive results are in white. As you can see, apart from a positive performance interval between midnight and 4 a.m., a breakout approach does not perform at all. Conversely, fake breakout strategies seem to have significantly more chances of success on such a cross.

This result encourages us to complete the development process for a fake breakout strategy in EURGB.

To backtest a fake breakout strategy, let's replace the following block code:

```
if time>=SR_Time and SR_count<Waitbars then begin
    if close < Resistance then buy       next bar 100000
    shares  Resistance stop;
    if close > Support    then sellshort next bar 100000
    shares  Support stop;
End;
```

We'll replace it with this new code:

```
if time>=SR_Time and SR_count<Waitbars then begin
    if close < Resistance then sellshort       next bar 100000
    shares  Resistance limit;
    if close > Support    then buy             next bar 100000
    shares  Support    limit;
End;
```

As you can see by comparing the two block codes, the entry conditions have been reversed; the breakout strategy opens a long position when the **`Resistance`** level has been broken, using a stop order, whereas the fake breakout strategy opens a short position when the resistance level has been reached, using a limit order.

Conversely, the breakout strategy opens a short position when the **`Support`** level has been broken by using a stop order, whereas the fake breakout strategy opens a long position when the **`Support`** level has been touched by using a limit order.

In *Figure 8.19*, you find the complete input list.

TradeStation Strategy Performance Report - EURGBP 60 min (14/04/2024-05/07/2024)
Account (USD)
ALL Data
TradeStation Chart Settings
Symbol EURGBP
Description Euro / British Pound
Interval 60 min.
Start Date/Time 14/04/2024 18:00:00
End Date/Time 05/07/2024 16:59:00
TradeStation Strategies Applied
Chapter8 Forex Fake BreakOut(On)
TradeStation Strategy Inputs
Chapter8 Forex Fake BreakOut - SR_Time 900
Chapter8 Forex Fake BreakOut - SR_BarsBack 24
Chapter8 Forex Fake BreakOut - WaitBars 6
Chapter8 Forex Fake BreakOut - AtrLen 120
Chapter8 Forex Fake BreakOut - StopAtrs 2
Chapter8 Forex Fake BreakOut - RewRiskRatio 3
Performance Summary
Trade Analysis
Trades List
Periodical Returns
Performance Graphs
Trade Graphs
Settings

Figure 8.19 – The EURGBP fake breakout strategy settings

As you see in *Figure 8.19*, we have a volatility money management based on 120 bars Atr, 2 **`StopAtrs`**, and 3 **`RewRisk`**.

> ***NOTE***
>
> *The **`RewRiskRatio`** (reward divided by risk) input is a very convenient way to impose a personalized risk-reward ratio on the strategy: for example, **`RewRisk=1`** means that the strategy's stop loss will be equal to the target; **`RewRisk=2`** means that the strategy's target will be twice the stop loss.*

In *Figure 8.20*, you can find some entry/exit examples:

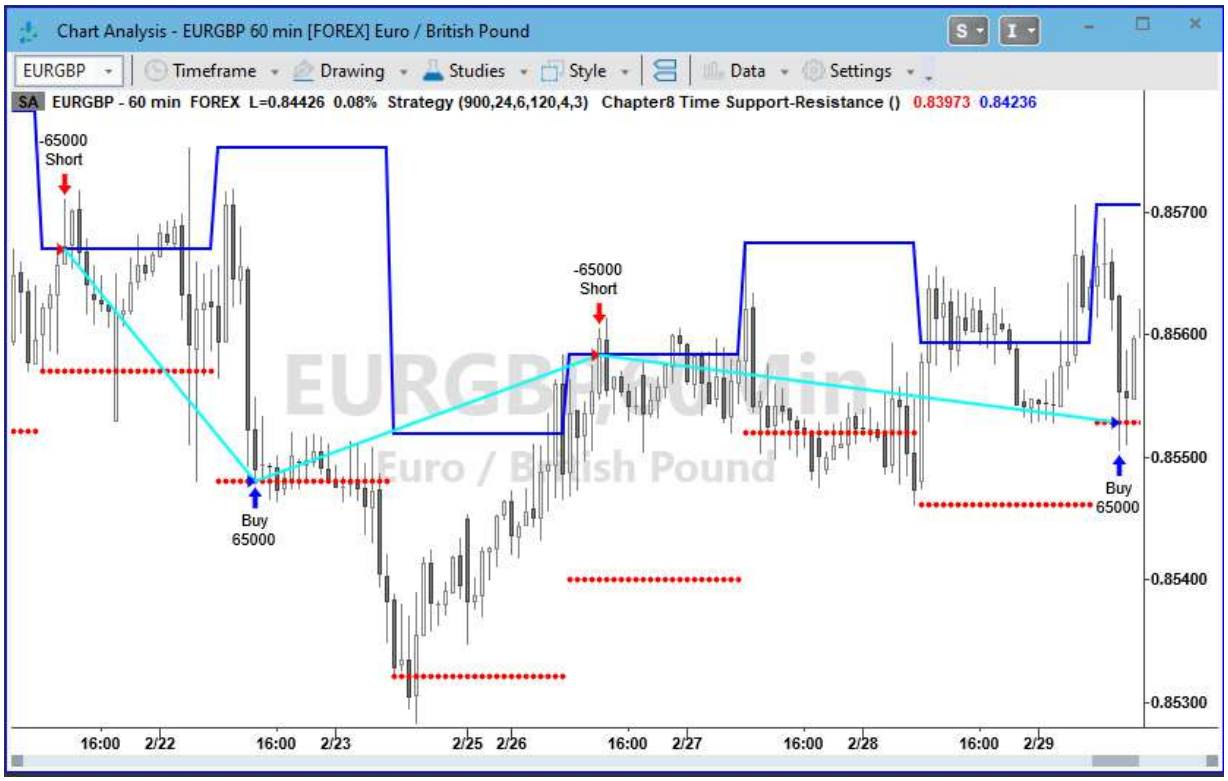


Figure 8.20 – The EURGBP fake breakout strategy – entry/exit examples

As you see in *Figure 8.20*, the strategy opens a new trade when the price reaches a support or a resistance level, working in exactly the opposite way to the breakout approach. In *Figure 8.21*, you can see the related equity line:

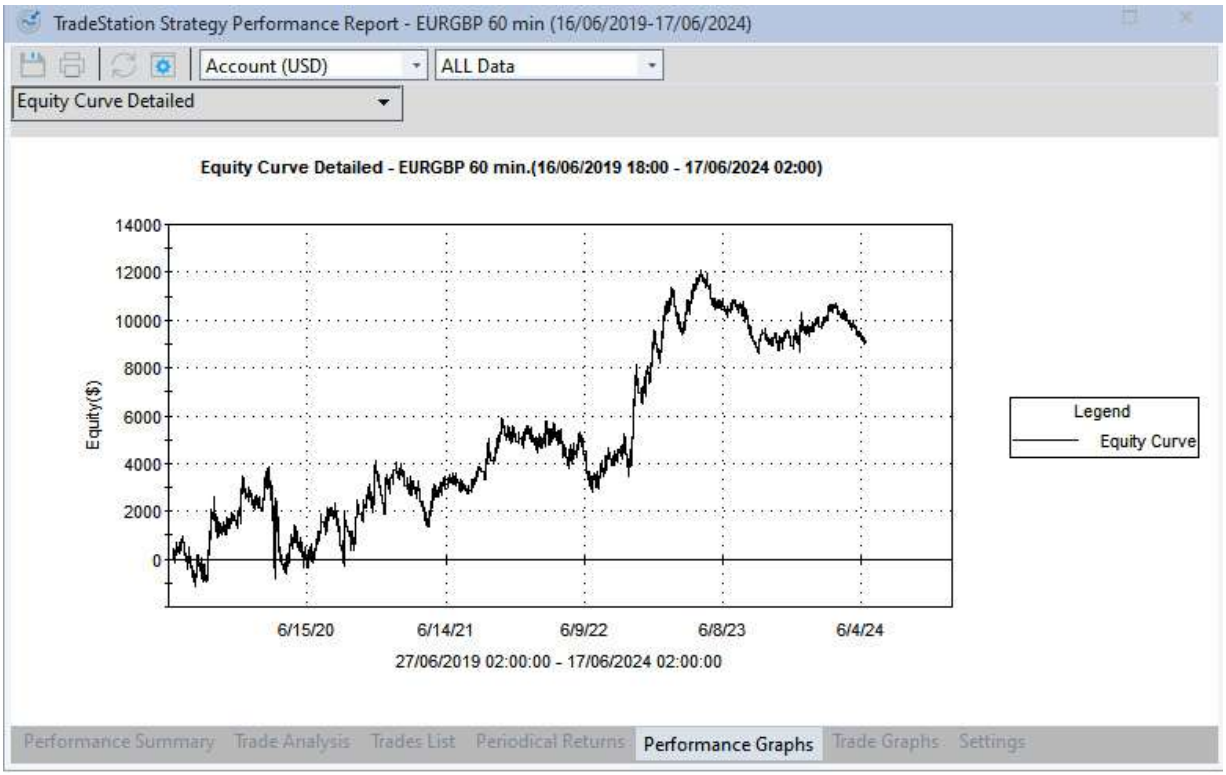


Figure 8.21 – The EURGBP fake breakout strategy equity line

As you see in *Figure 8.21*, a fake breakout approach applied to EURGBP shows a profit and a growing equity line.

> ***NOTE***
>
> *Remember that all the strategies you find in this book, although derived from real-world trading, are presented for educational purposes only and cannot be traded as they are. Instead, they should be used as a starting point for analysis and inspiration to create your own personalized strategies.*

In this section, we delved into the forex market, a very complex subject that requires the study of fundamental concepts such as interest rates and 24-hour trading sessions. We have seen that, unlike the stock market, the forex market can easily move in both directions, long and short. We explored the differences between breakouts and fake breakouts and realized that not all currencies are directional. In fact, in some cases, such as EURGBP, a fake breakout approach makes more sense, whereas, in other crosses, such as USDJPY, a breakout strategy is more appropriate. Then, we built some algorithms and used them to conduct sensitivity analysis, studying the markets through the TradeStation backtesting tools.

Summary

In this chapter, we studied the basic characteristics of the futures and forex markets, highlighting the main differences compared to the stock market, and learned how to handle such financial assets through TradeStation. We then proposed some algorithms suitable for both analysis and algorithmic trading and backtested them, using EasyLanguage-automated strategies. Remember, as we discussed in *Chapter 1*, that algorithmic trading and automated trading are not the same, as trading algorithms can be programmed with varying levels of automation. This will be the subject of the next chapter, which focuses on trading plans.

9

The Trading Operational Plan

So far, we have approached the topic of algorithmic trading at a theoretical level. We have explained the basic functioning of the main financial markets, written algorithms, and tested them to understand their robustness. Imagine you have completed all the tasks described so far and have your own algorithmic strategy ready to go to market. In this chapter, we will try to describe how to move from theory to practice and how to organize algorithmic resources to achieve our trading objectives. By the end of this chapter, you will be aware of what a trading plan is, which is the better level of automation for your trading style, and how TradeStation and EasyLanguage can help you in real-life market trading.

In this chapter, we are going to discuss the following:

- What is an operational plan?
- A fully automated trading plan for futures
- A semi-automated trading plan on large stock lists

Remember that putting into practice the advice and guidelines provided by a book, such as the one you are reading, is crucial for building your market operations. The theories you have learned so far will eventually need to face the realities of the markets. My advice is to avoid implementing the strategies you develop with real money before applying them to your simulated account. Inevitably, a novice trader loses money in the real market, so you must do everything possible to preserve your capital. TradeStation provides the opportunity to use a simulated account that is very close to reality, and I encourage you to use it.

What is an operational plan?

An operational plan is a detailed, actionable framework designed to achieve specific goals through the coordination and deployment of resources. While the concept is critical in the military, it is also widely applicable in various non-military fields, such as business and investments. In a trading or investment-related environment, an operational plan mainly deals with managing resources, such as in *Figure 9.1*:

Figure 9.1 – Trading operational plan resources

A **trading platform** is a software application that allows traders and investors to place trades and manage their accounts in the financial markets. It serves as the interface between the trader and the market, providing the necessary tools and resources to execute trades, analyze market data, and manage trading strategies.

Market data refers to information and statistics about current and historical prices, trading volumes, and other relevant metrics for financial instruments traded in various markets. Market data is essential for traders and investors to analyze market trends, make trading decisions, and manage their portfolios effectively.

At this point in the book, you should have an idea of what is meant by **algorithms** in the context of an operational trading plan.

Brokers play a pivotal role in algorithmic trading by providing access to financial markets and executing trades on behalf of clients.

Some trading projects might require a **team**; in this case, each team member might need algorithms to assist them in their tasks. For example, operators responsible for execution need algorithmic tools for automated entries/exits, while analysts need tools such as scanners and RadarScreen.

An operational plan can have various levels of flexibility. For example, a **tight** plan has very rigid rules and guidelines that must be followed exactly. It ensures consistency but may lack adaptability to changing market conditions. Conversely, a **loose** plan offers broad guidelines and allows for significant discretion based on the trader's judgment. It enables adaptability and responsiveness to market changes but requires a high level of expertise to be managed effectively.

In the next section, we will explore how to manage a very tight operational plan by presenting an example of a fully automated strategy to trade four futures.

A fully automated trading plan for futures

As you will remember, in *Chapter 1*, we discussed the different levels of automation in algorithmic trading. In this section, we are going to provide instructions to start a real fully automated trading

plan on a four-futures portfolio. For educational purposes, in this chapter, we will apply a time breakout strategy to the following futures:

- **E-mini S&P 500**: This represents the US standard S&P 500 index
- **Gold**: The precious metal that is often traded as a commodity and used as a safe-haven investment during times of economic uncertainty
- **T-note**: The marketable US government debt security, with a fixed interest rate and a maturity between 1 and 10 years
- **Yen**: The official currency of Japan against the US dollar

A trading strategy incorporating E-mini S&P 500, gold, T-notes, and yen offers significant advantages due to the typically low correlations among these assets; during risk-on periods, investors prefer equities (E-mini S&P 500) and higher-yielding assets, leading to a rise in stock prices and potentially weaker gold, T-notes, and yen.

During risk-off periods, investors seek safe-haven assets such as gold, T-notes, and yen, resulting in a decline in stock prices (E-mini S&P 500) and a rise in the prices of these safe-haven assets.

Conversely, the correlations among gold, T-notes, and yen can vary significantly over time, depending on various factors, such as inflation, interest rate decisions, and geopolitical events.

In this section, we are going to learn about the following:

- How to manage futures symbols for real trading
- How to size a four-futures portfolio
- How to manage live fully automated strategies

How to manage futures symbols for real trading

In *Chapter 8*, we learned how to deal with TradeStation futures symbols for backtesting. In order to live-execute orders on futures contracts, we need to properly identify the contract to trade. In *Table 9.1*, you can see the TradeStation symbols for live trading:

Futures contracts	**Continuous contract symbol**	**Symbol for trading**	**Continuous contract symbol for trading**
E-mini S&P 500	@ES	ESU24	@ESU24
Gold future	@GC	GCQ24	@GCQ24
T-note future	@TY	TYU24	@TYU24

Yen future	@JY	JYU24	@JYU24

Table 9.1 – The futures list

In *Table 9.1*, you can see the list of futures contracts we are assuming to start trading live. Note that in order to perform real algorithmic trading, you need to properly identify the futures contract, including the expiration date. TradeStation offers a couple of alternatives. For example, to trade the E-mini S&P 500 futures contract, you can use the **ESU24** symbol, which identifies the front contract (expiring in September 2024) as of today's date (it is June 2024 at the time of writing). Alternatively, you can use the **@ESU24** symbol, which represents a continuous futures contract and allows you to monitor historical trades on a chart.

How to size the four futures portfolio

As you will recall from *Chapter 7*, we learned that two of the key aspects of algorithmic trading are **risk management** and **position sizing**. Position sizing is even more relevant in trading with futures, where the sizes are expressed in the number of contracts, and you cannot fraction the quantities below the minimum value of one contract. Therefore, it would be a good idea, before activating our strategies, to understand the maximum risk we are willing to take and, especially, what sizes we trade our four strategies with.

In order to achieve such tasks, in my experience, I have found it very useful to use TradeStation and Excel together. In fact, even though TradeStation has a very powerful tool for portfolio backtesting, Portfolio Maestro, I found more flexibility in exporting data from Print Log and building my own Excel portfolio simulator. Let's see how to do that.

First of all, let's modify the time breakout strategy we used in *Chapter 8* by adding a few lines:

```
input:Direction(0),Size(1),SR_Time(800),SR_BarsBack(12),WaitBars(1);
var:Resistance(0),Support(0),SR_count(0);
if time=SR_Time then Begin
    Resistance=highest(H,SR_BarsBack);
    Support=lowest(L,SR_BarsBack);
    SR_count=0;
End;
SR_count=SR_count+1;
//Entry
if time>=SR_Time and SR_count<Waitbars then begin
    if Direction>=0 then buy       Size shares next bar Resistance
    stop;
    if Direction<=0 then sellshort Size shares next bar Support stop;
End;
Input:BarsIN(24);
if BarsIN>0 and barssinceentry = BarsIN then Begin
    if mp = 1 then sell next bar open;
    if mp =-1 then buytocover next bar open;
End;
if date<>date[1] then print(symbol,",",date,",",Netprofit+Openpositionprofit)
```

In this code, I added a **`Size`** input because we are going to trade different sizes for different assets; then, in order to populate Print Log, I added the following lines:

```
if date<>date[1] then print(symbol,",",date,",",Netprofit+Openpositionprofit)
```

In *Figure 9.2*, you can see how the TradeStation workspace looks when we applied the time breakout on the four selected futures:

Figure 9.2 – The time breakout strategy applied on TY-GC-JY-ES 60 min charts

As you can see from *Figure 9.2*, the script will populate Print Log with data that you can copy and paste into the Excel portfolio simulator, as shown in *Figure 9.3*.

	A	B	C	D	E	F	G	H	I	J	K
1	Symbol	Date	Equity	Size	Equity_rec						
2	@TYU24	1190709	140.62	1	140.62						
3	@TYU24	1190710	-187.5	1	-187.5						
4	@TYU24	1190711	250	1	250		*Formula in D2*	=VLOOKUP(A2;Portfolio!$H$19:$I$22;2)			
5	@TYU24	1190712	531.25	1	531.25						
6	@TYU24	1190714	406.25	1	406.25		*Formula in E2*	=C2*D2			
7	@TYU24	1190715	546.88	1	546.88						
8	@TYU24	1190716	578.12	1	578.12						
9	@TYU24	1190717	437.5	1	437.5						
10	@TYU24	1190718	718.75	1	718.75						
11	@TYU24	1190719	359.38	1	359.38						
12	@TYU24	1190721	531.25	1	531.25						
13	@TYU24	1190722	500	1	500						
14	@TYU24	1190723	312.5	1	312.5						
15	@TYU24	1190724	-125	1	-125						
16	@TYU24	1190725	46.88	1	46.88						
17	@TYU24	1190726	234.38	1	234.38						

Portfolio | PrintLog

Figure 9.3 – Excel populated by TradeStation Print Log

By using a PivotTable on the data in *Figure 9.3* and adding some simple formulas, we can easily integrate the Excel portfolio simulator with TradeStation, as shown in *Figure 9.4*.

> ***NOTE***
>
> *If you are not familiar with Excel formulas, you will find the Excel portfolio simulator in the GitHub repository, available at https://github.com/PacktPublishing/Tradestation-Easylanguage-for-Algorithmic-Trading.*

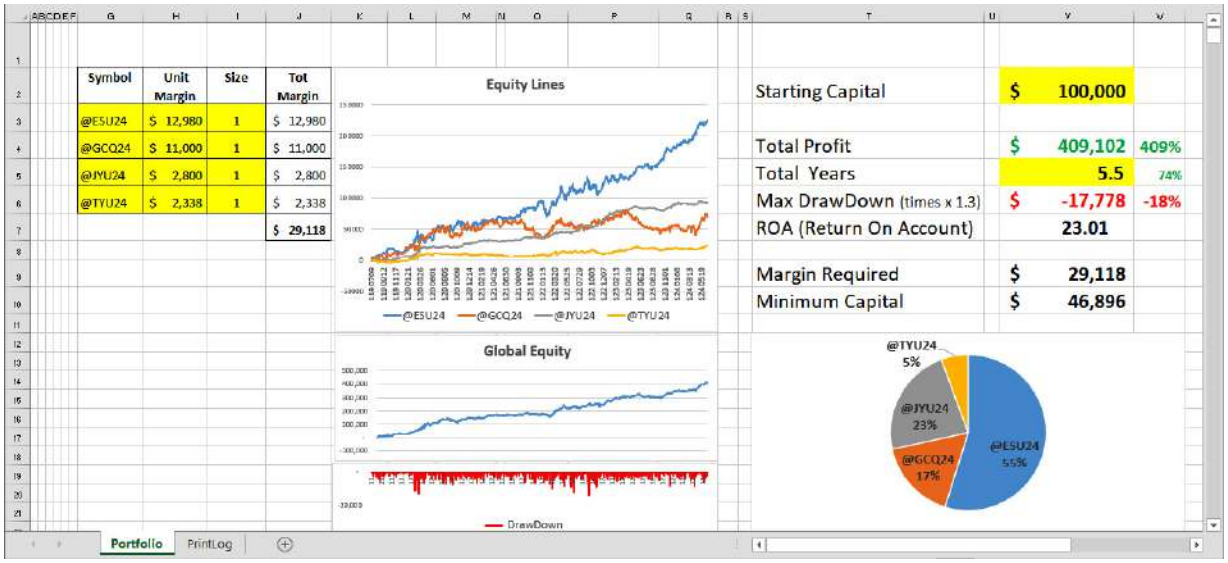


Figure 9.4 – The Excel portfolio simulator with one contract of ES-GC-TY-JY

In *Figure 9.4*, you can see the Excel portfolio simulator integrated with TradeStation. This tool allows us to simulate the effects of changes in sizing on key performance indicators, such as total profit, maximum drawdown, and **Return on Account** (**ROA**). In this example, we simulate the allocation of $100,000 capital to time breakout strategies with equal sizing, meaning one contract for each of the four selected futures.

As shown, the entire portfolio exhibits a total profit of $409,102, a maximum drawdown of -$17,778, and an ROA of 23.01. While these results are quite favorable, the pie chart reveals a skew toward @ESU24, which accounts for 55% of the total profits, whereas @TYU24 only contributes 5%. This tool allows us to answer questions such as, what happens to the portfolio ROA if we reduce the size of @ESU24, increase the size of @TYU24, or both?

Without relaunching TradeStation strategies, you can simply adjust the sizes in the Excel simulator and observe the results. In *Figure 9.5*, you can see the Excel portfolio simulator after we reduced the size of ES from 1 contract to 0.5, and then refreshed the PivotTable.

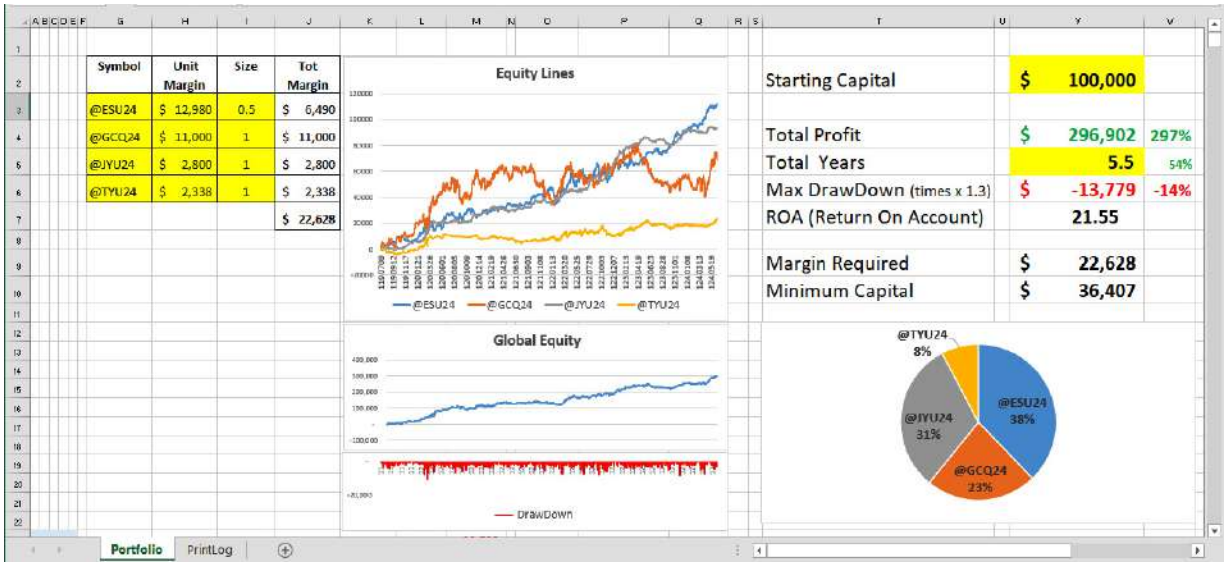


Figure 9.5 – Excel portfolio simulator – the ES size = 0.5

As you see in *Figure 9.5*, the portfolio performance has been recalculated, showing a total profit of $296,902, a maximum drawdown of -$13,779, and an ROA of 21.55. The pie chart indicates that the performance distribution is now more balanced, although the TY performance weight remains low at 8%.

To further explore the impact of sizing adjustments, let's increase the number of TY contracts from 1 to 5. After updating the TY size in the Excel simulator and refreshing the PivotTable, we can observe the resulting portfolio performance in *Figure 9.6*.

> **NOTE**
>
> *The E-mini S&P 500 future is also available in the micro size variant, which is 1/10th of the standard contract size, providing traders with a more accessible and flexible trading option. This is the reason it makes sense to simulate a one half contract, which equals five micro futures contracts.*

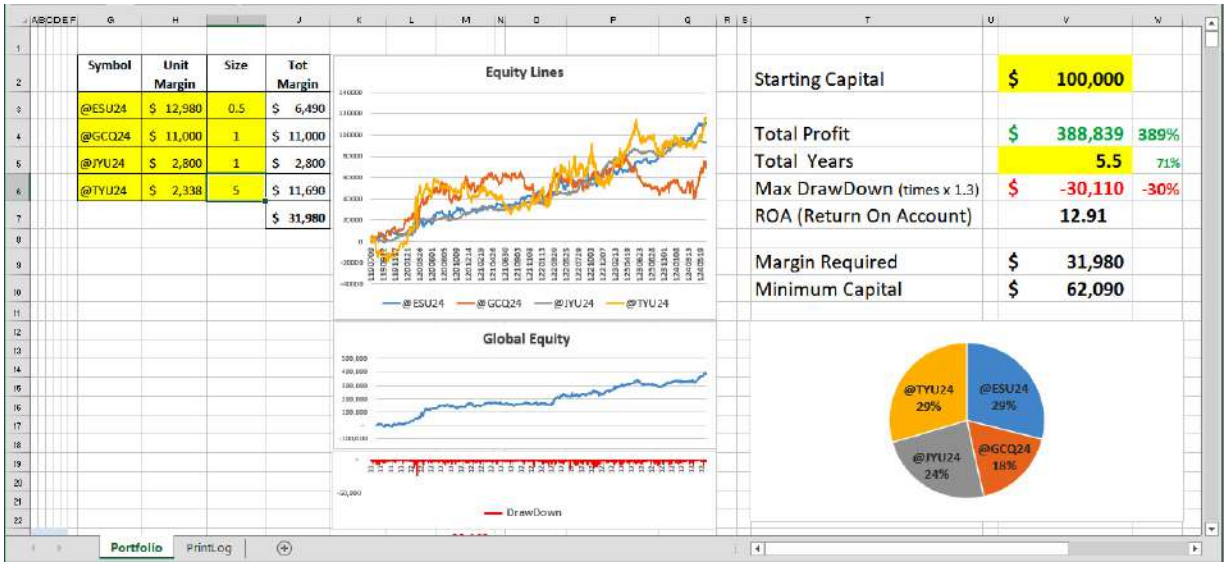


Figure 9.6 – The Excel portfolio simulator – the ES size = 0.5 and the TY size = 5

Figure 9.6 shows a total profit of $388,839, a maximum drawdown of -$30,110, and an ROA of 12.91. While the performance weights for the four futures are now balanced, the decrease in ROA might suggest that trading five contracts of TY may be excessive.

As you can see, the Excel portfolio simulator is a very flexible tool and allows us to answer three questions:

- What is the minimum capital required to execute this strategy on four futures?
- If you trade your own money, is the risk-reward profile provided by the portfolio simulator in line with your risk attitude?
- If you trade someone else's money, is the risk-reward profile in line with your client's risk attitude?

Minimum capital required

Answering the first question is quite easy, as the formula to calculate the minimum capital required is as follows:

Minimum Capital Required =

Total Margin Required + Maximum Historically Drawdown (increased by 30%)

The portfolio simulator calculates and updates the formula shown in *Table 9.2*. So, for example, the simulator, as shown in *Figure 9.6*, tells us that the minimum capital required is $62,090, which is the

summation between the margin required ($31,980) and the maximum historical drawdown ($30,110) USD.

Risk management

The answers to the second and third questions require you to carefully consider your (and your client's) risk tolerance. Specifically, can you endure a drawdown of $30,000 without experiencing undue stress, while maintaining the clarity and motivation necessary to continue operating even during temporary setbacks?

You need to be very honest in answering this question because the success of the entire project depends on how you will manage your business during the inevitable market storms you are going to face. Most of the failures I have witnessed during my experience were caused by not clearly answering (and sharing with all the stakeholders involved in the trading project) a very simple question – what is the maximum drawdown endurable without feeling distressed?

If you realize that the risk profile coming from the portfolio simulator is too high for your or your client's risk tolerance, you have to return to the simulator to review the size, or even revisit the strategy development. Do anything necessary to maintain risk within given boundaries *before* a project kicks off. Once the games have begun, it will be impossible to turn back.

Very often, junior traders attending my training courses ask me questions such as, if risk management is so important in algorithmic trading, why do you frequently use strategies that only have a time-based exit and no money management rules? There are several reasons why I believe that time-based exits are the best for algorithmic trading:

- By adding highly elaborate money management rules, the system's degrees of freedom increase, which can lead to an out-of-sample temporal decay.
- Risk management, and especially position management, is a highly personal aspect of trading. This is why I prefer not to provide traders with strict guidelines on these matters.
- A time-based exit, whether after a few days or weeks, is an effective way to control risk, especially when thorough backtesting is conducted.
- In strategies such as the one presented on the four futures markets, since the markets are often decorrelated, they tend to balance each other out, especially if the position sizes are well-weighted.

In this section, we explored the method of precisely determining position sizes through an Excel portfolio simulator integrated with TradeStation. This approach enables us to customize risk management strategies to align with the specific risk preferences of individual traders. By using this tool, traders can optimize their portfolio allocations effectively, ensuring that they adhere to their desired level of risk exposure.

Let's now move on to how to set up TradeStation for live trading.

How to manage live fully automated strategies

In order to activate live trading, from the **Chart** menu, go to **Studies** | **Edit Strategies**. A **Customize Studies & Strategies** dialog will open, as shown in *Figure 9.7*.

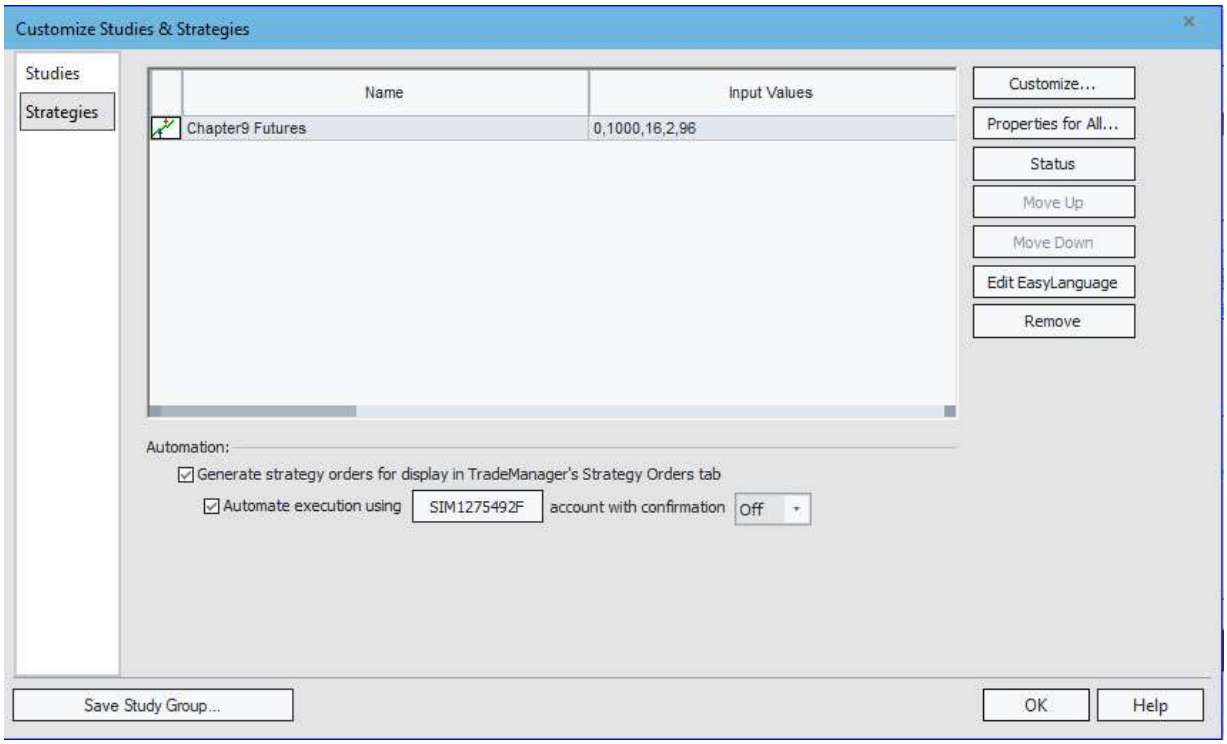


Figure 9.7 – Customize Studies & Strategies

In *Figure 9.7*, you can see how to set the strategy for live trading. Once done, at the top left of the chart, the **SA** icon will be colored red with a dark green background, meaning that the strategy is trading live with **confirmation OFF** (*Figure 9.8*).

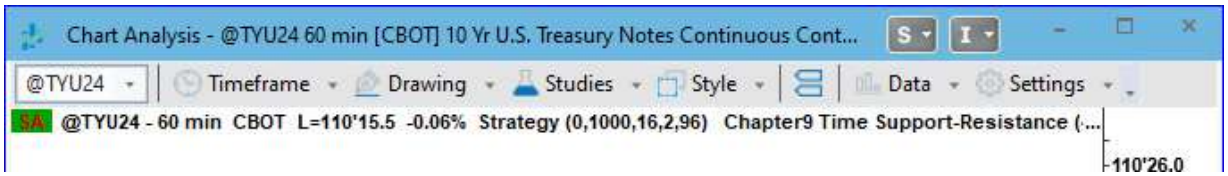


Figure 9.8 – Charts ready to trade live with conformation OFF

The SA icon at the top left in *Figure 9.8* will turn from red to yellow if you select **confirmation ON**.

You need to repeat the same procedures for the four futures charts. But how do you monitor strategies?

In *Figure 9.9*, you can see the strategy positions selected from the **TradeManager** menu.

TradeManager - Strategy Positions
Data
Settings
Symbol Filter
Account No
All
Symbol
Interval
Strategy Position
Open Position
Positions Match
Strategy Price
Strategy P/L
1 TYU24 60 min. Short 1 Flat False $110.36 $718.75
2 JYU24 60 min. Flat Flat True
3 GCQ24 60 min. Long 1 Flat False $2,340.50 ($360.00)
4 ESU24 60 min. Short 1 Flat False $5,556.75 $1,225.00
Total strategy profit and loss is $1,583.75
NOTE: The "Positions Match" column is a general indicator of whether or not your actual open position in ...
NOTE: The list is automatically sorted every minute. Last sort on 30/06/2024 at 18:05:28
Orders
Average Price
Staged Orders
Positions
Strategy Orders
Strategy Positions

Figure 9.9 – TradeManager set to show strategy positions

To open TradeManager as shown in *Figure 9.9*, from the main menu, go to **File** | **New Application** | **TradeManager**.

If we observe the `Open Position` column in *Figure 9.9*, we can see that there is no position opened; by observing the `Positions Match` column, we can see at a glance that the only synchronized position is JY because it is the only one showing **True**. This facilitates a trader's monitoring and allows them to intervene quickly to correct any errors.

> ***NOTE***
>
> *TradeManager is very useful for understanding what is going on with your automated trading at a glance. However, it is strongly recommended not to leave the platform unattended while trading automatically.*

In this section, we took a look at how to manage a very tight operational plan of four decorrelated futures. Let's now move on to a looser plan, a semi-automated approach to trade stocks.

A semi-automated trading plan on large stock lists

After understanding how a tight plan with fully automated futures works, we will now move on to managing a looser semi-automated plan on stock lists.

In my experience, the stock market, particularly the US market, yields optimal trading results in daily, weekly, or even monthly timeframes. Indeed, within mature markets (such as the US, Europe, and the UK), the trading environment is too crowded for intraday strategies to consistently succeed, although this viewpoint is based on personal experience.

This suggests that algorithms used for stocks do not necessarily require real-time trading. This approach not only proves more effective over the long term but also allows traders or management teams to avoid high-stress situations associated with rapid execution and overtrading, which can undermine trading success both psychologically and financially.

In this section, we will create a semi-automated trading plan for trading across a broad list of large stock lists, using our trend and pullback strategy. So, before we begin, take some time to refer back to *Chapter 6*.

In this section, we are going to understand the following:

- How to track entries with RadarScreen
- How to calculate volatility-based sizes
- How to monitor real positions

How to track entries with RadarScreen

Let's start by opening a new RadarScreen page from the main menu by going to **File** | **New Application** | **RadarScreen**. Then, in order to populate the grid with Nasdaq-100 components, from the **RadarScreen** menu, go to **Data** | **Add Symbol List** | **Nasdaq 100**.

If you recall, in the previous chapters, we discussed TradeStation RadarScreen, a tool that enables you to analyze large lists of financial instruments on a single page. At the core of the indicators for RadarScreen are the same algorithms we programmed in *Chapters 3*, *4*, *5*, and *6*. The only difference lies in the output – chart indicators draw lines or histograms on a single symbol's chart for the entire available historical period, whereas RadarScreen indicators signal the presence of a pattern in the latest historical bar for all symbols entered in a grid.

So, for example, we can rewrite the trend and pullback strategy we programmed in *Chapter 6*, as shown in the following RadarScreen indicator:

```
Input: FastLen(20),SlowLen(200),RsiLen(5),RsiTrigger(30);
var:Trend(false),Pullback(false),Signal(false),SignalCount(0),Entry_(0),PerfPct(0);
Trend    = average(close,FastLen) > average(Close,SlowLen);
Pullback = rsi(close,RsiLen) cross over RsiTrigger;
Signal   = Trend and Pullback;
if Signal then SignalCount=1 else SignalCount=SignalCount+1;
if Signal then Entry_=close;
setplotbgcolor(1,white);
setplotbgcolor(2,white);
setplotbgcolor(3,white);
setplotbgcolor(4,white);
if Trend    then begin
    plot1("UP","Trend");
    setplotbgcolor(1,rgb(251,234,3));
End;
if Signal   then begin
    plot2("PullBack","Pullback");
```

```
    setplotbgcolor(2,rgb(218,228,155));
End;
if Signal   then begin
    plot3("Possible BUY","Signal");
    setplotbgcolor(3,green);
End;
if SignalCount>0 and Trend then begin
    plot4(numtostr(SignalCount,0),"Days since last Signal");
    setplotbgcolor(4,green);
    if entry_<>0 then plot5(close/entry_-1,"Perf% since last
    Signal",darkgreen);if plot5<0 then setplotcolor(5,red);
End;
```

As you can see in the preceding code, in order to detect **`Trend`**, we used two **Simple Moving Averages** (**SMAs**). In order to detect **`Pullback`**, we used an RSI crossover, and in order to detect **`Signal`**, we used a condition where both the **`Trend`** and **`Pullback`** conditions were **`True`**. This is exactly the same as the logic we wrote in *Chapter 6*.

In *Figure 9.10*, you can see what this looks like on RadarScreen:

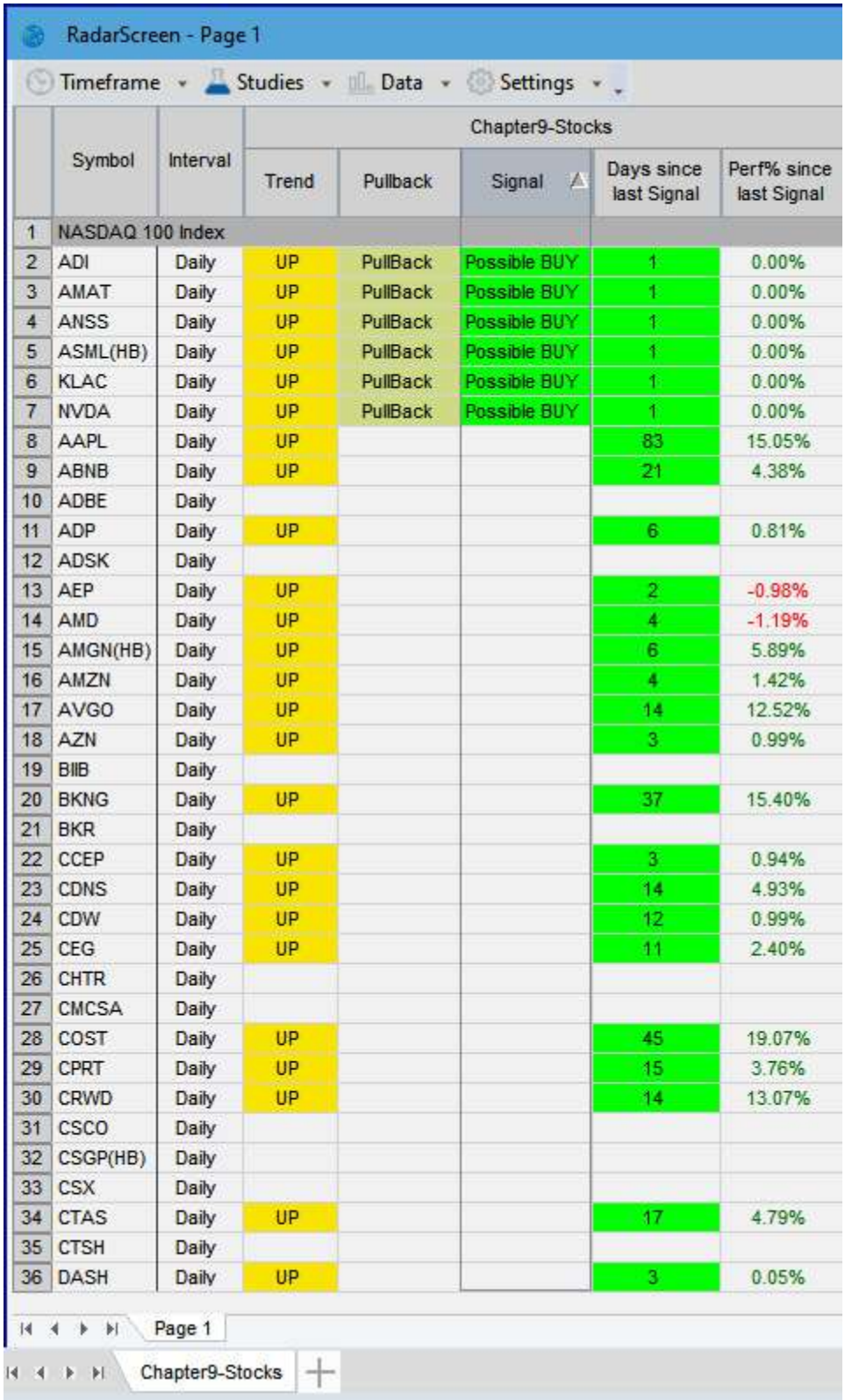

RadarScreen - Page 1
Timeframe
Studies
Data
Settings
Chapter9-Stocks
Symbol
Interval
Trend
Pullback
Signal
Days since last Signal
Perf% since last Signal
1 NASDAQ 100 Index
2 ADI Daily UP PullBack Possible BUY 1 0.00%
3 AMAT Daily UP PullBack Possible BUY 1 0.00%
4 ANSS Daily UP PullBack Possible BUY 1 0.00%
5 ASML(HB) Daily UP PullBack Possible BUY 1 0.00%
6 KLAC Daily UP PullBack Possible BUY 1 0.00%
7 NVDA Daily UP PullBack Possible BUY 1 0.00%
8 AAPL Daily UP 83 15.05%
9 ABNB Daily UP 21 4.38%
10 ADBE Daily
11 ADP Daily UP 6 0.81%
12 ADSK Daily
13 AEP Daily UP 2 -0.98%
14 AMD Daily UP 4 -1.19%
15 AMGN(HB) Daily UP 6 5.89%
16 AMZN Daily UP 4 1.42%
17 AVGO Daily UP 14 12.52%
18 AZN Daily UP 3 0.99%
19 BIIB Daily
20 BKNG Daily UP 37 15.40%
21 BKR Daily
22 CCEP Daily UP 3 0.94%
23 CDNS Daily UP 14 4.93%
24 CDW Daily UP 12 0.99%
25 CEG Daily UP 11 2.40%
26 CHTR Daily
27 CMCSA Daily
28 COST Daily UP 45 19.07%
29 CPRT Daily UP 15 3.76%
30 CRWD Daily UP 14 13.07%
31 CSCO Daily
32 CSGP(HB) Daily
33 CSX Daily
34 CTAS Daily UP 17 4.79%
35 CTSH Daily
36 DASH Daily UP 3 0.05%
Page 1
Chapter9-Stocks

Figure 9.10 – Trend and Pullback on RadarScreen

As you can see in *Figure 9.10*, the indicator displays five columns:

- The **`Trend`** column, which may show **UP** or remain empty
- The **`Pullback`** column, which may show **PullBack** or remain empty
- The **`Signal`** column, which highlights **Possible BUY** or remains empty
- The **`Days since last Signal`** column, indicating how many days ago the last signal appeared
- The **`Perf% since last Signal`** column, showing the stock's net change since the last signal

Once we have identified the potential stocks to buy from those listed on RadarScreen, the next immediate question is, how much do we buy?

How to calculate volatility-based sizes

As you will remember from *Chapter 6*, in a proper risk management approach, in order to set the trading size, we need to take into consideration individual volatility. In the following script, there is an algorithm to calculate size according to the stock's beta value, which measures a stock's volatility in relation to the market:

```
Input: BaseCapital(10000),MinDollarSize(4000),MaxDollarSize(20000);
variables:
    BetaValue( 0 ),
    oErrorCode( 0 );
oErrorCode = 0;
BetaValue = FundValue( "BETA", 0, oErrorCode );
var:TradeSizeUSD(0),TradeSizeShares(0);
if BetaValue <>0 then begin
    TradeSizeUSD    = maxlist(BaseCapital/BetaValue,MinDollarSize);
    TradeSizeUSD    = minlist(TradeSizeUSD,MaxDollarSize);
    TradeSizeShares = TradeSizeUSD / Close;
End;
plot1(BetaValue,"Beta");
plot2(TradeSizeUSD,"Trade Size (USD)");
plot3(TradeSizeShares,"Trade Size (shares)");
```

The objective of the aforementioned algorithm is to automatically calculate the trade sizes of all the stocks listed in RadarScreen, by weighting them with their beta.

> ***NOTE***
>
> *In the preceding code, we retrieve the beta value for each stock by using the **`FundValue( "BETA", 0, oErrorCode );`** function. The **`FundValue`** functions are built-in TradeStation functions that provide a long list of basic information. You can find more information in the **`EasyLanguage Help`** file.*

In *Figure 9.11*, you can see how RadarScreen looks after we added the preceding indicator:

	Symbol	Chapter9-Sizes		
		Beta	Trade Size (USD)	Trade Size (shares)
1	NASDAQ 100 Index			
2	AZN	0.47	20,000	256
3	GILD	0.20	20,000	291
4	REGN	0.13	20,000	19
5	VRTX	0.39	20,000	42
6	XEL(HB)	0.38	20,000	371
7	EXC	0.51	19,763	570
8	TMUS	0.51	19,718	111
9	KHC	0.52	19,413	602
10	AEP	0.52	19,090	216
11	PEP	0.53	19,041	115
12	MDLZ	0.54	18,584	279
13	AMGN(HB)	0.58	17,236	55
14	PDD	0.61	16,421	124
15	KDP	0.62	16,144	473
16	TEAM	0.74	13,436	78
17	MNST	0.76	13,210	261

Figure 9.11 – RadarScreen – volatility-based sizes

In *Figure 9.11*, you can see the volatility-based sizes for each stock on RadarScreen. Let's explain how to use it.

Assume you have a total capital of $100,000 and you want to invest it in 10 stocks. In a theoretical scenario where you choose 10 stocks with a beta of 1, it is correct to simply divide the total capital by 10 and buy $10,000 worth of each stock.

In a more common scenario where the stocks you choose have different betas, your trade size should be proportional to the beta. Therefore, it will be higher than $10,000 in the case of a beta lower than 1, and it will be lower than $10,000 in the case of a beta higher than 1.

How to monitor real positions

A very easy way to monitor real-time positions is TradeStation's TradeManager (*Figure 9.12*). You can access it from the main menu by going to **File** | **New Application** | **TradeManager** | **Positions**.

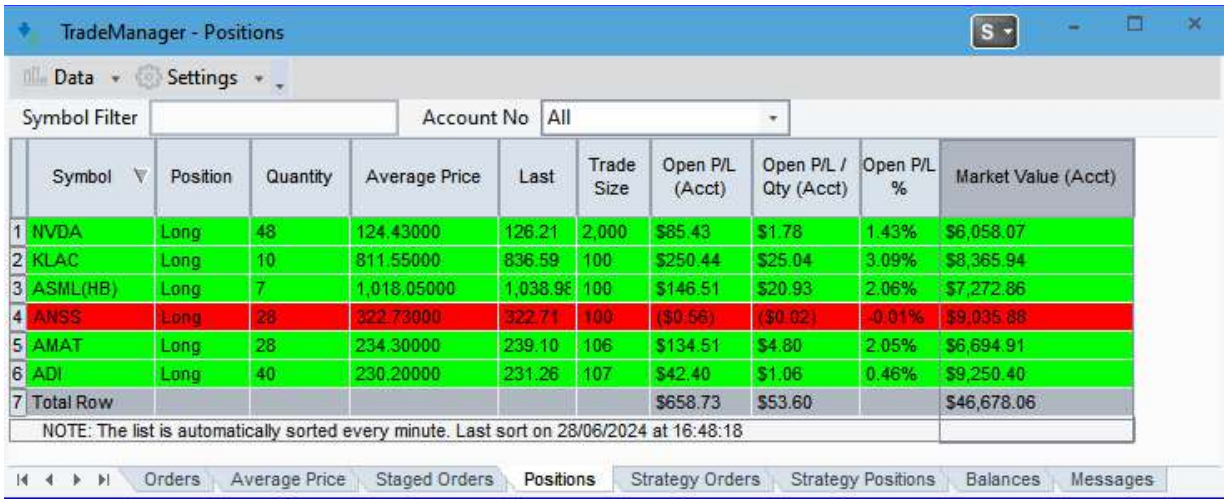

TradeManager - Positions
Data
Settings
Symbol Filter
Account No
All
Symbol
Position
Quantity
Average Price
Last
Trade Size
Open P/L (Acct)
Open P/L / Qty (Acct)
Open P/L %
Market Value (Acct)
1 NVDA Long 48 124.43000 126.21 2,000 $85.43 $1.78 1.43% $6,058.07
2 KLAC Long 10 811.55000 836.59 100 $250.44 $25.04 3.09% $8,365.94
3 ASML(HB) Long 7 1,018.05000 1,038.98 100 $146.51 $20.93 2.06% $7,272.86
4 ANSS Long 28 322.73000 322.71 100 ($0.56) ($0.02) -0.01% $9,035.88
5 AMAT Long 28 234.30000 239.10 106 $134.51 $4.80 2.05% $6,694.91
6 ADI Long 40 230.20000 231.26 107 $42.40 $1.06 0.46% $9,250.40
7 Total Row $658.73 $53.60 $46,678.06
NOTE: The list is automatically sorted every minute. Last sort on 28/06/2024 at 16:48:18
Orders
Average Price
Staged Orders
Positions
Strategy Orders
Strategy Positions
Balances
Messages

Figure 9.12 – TradeManager - Positions

In *Figure 9.12*, you can see TradeStation's TradeManager showing the open positions. Additionally, in order to monitor the account balance, you can select **Balances** from the **TradeManager** tab, as shown in *Figure 9.13*.

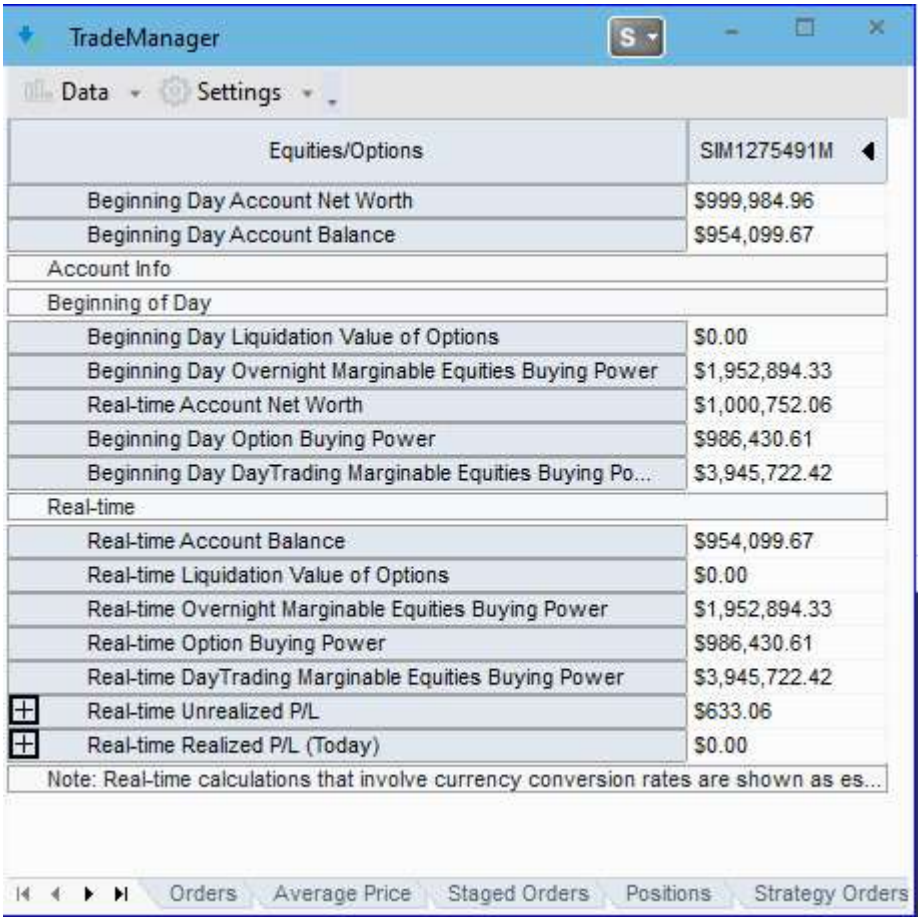

TradeManager
Data
Settings
Equities/Options
SIM1275491M
Beginning Day Account Net Worth $999,984.96
Beginning Day Account Balance $954,099.67
Account Info
Beginning of Day
Beginning Day Liquidation Value of Options $0.00
Beginning Day Overnight Marginable Equities Buying Power $1,952,894.33
Real-time Account Net Worth $1,000,752.06
Beginning Day Option Buying Power $986,430.61
Beginning Day DayTrading Marginable Equities Buying Po... $3,945,722.42
Real-time
Real-time Account Balance $954,099.67
Real-time Liquidation Value of Options $0.00
Real-time Overnight Marginable Equities Buying Power $1,952,894.33
Real-time Option Buying Power $986,430.61
Real-time DayTrading Marginable Equities Buying Power $3,945,722.42
Real-time Unrealized P/L $633.06
Real-time Realized P/L (Today) $0.00
Note: Real-time calculations that involve currency conversion rates are shown as es...
Orders
Average Price
Staged Orders
Positions
Strategy Orders

Figure 9.13 – TradeManager balances

Using TradeManager balances (*Figure 9.13*), you can gain a comprehensive view of your trading account, allowing you to monitor account balances and check margin requirements.

The final version of the TradeStation workspace for real stock trading is shown in *Figure 9.14*.

Figure 9.14 – The TradeStation workspace to trade for real

In *Figure 9.14*, you can see the TradeStation workspace displaying, from left to right, two RadarScreens windows (one for tracking entries and another for volatility-based quantities), one chart, and TradeManager positions and balances.

In this section, we have explored organizing TradeStation to execute an operational plan with stock lists. We observed that, unlike futures trading, stock trading is more suited to slower timeframes that do not necessarily require real-time market monitoring. We also discussed programming indicators for RadarScreen to identify entry timing and determine sizes proportionally, based on volatility.

Summary

In this chapter, we have dedicated our efforts to the operational trading plan, focusing on how to organize all the necessary resources to go to market. We have explored two practical examples – one showcasing total automation on four futures and the other demonstrating partial automation on stock lists. Through these examples, we highlighted the nuances of different trading strategies and the importance of adapting our approach based on the specific instruments being traded.

In the next chapter, we are going to delve into a new state-of-the-art frontier – a transformative approach that combines the predictive power of machine learning with the strategic insights of technical trading, opening up new possibilities to achieve trading success.

10

EasyLanguage in AI – Bridging Traditional Trading and Advanced Analytics

There is a lot of enthusiasm surrounding the use of **Artificial Intelligence** (**AI**) in the financial sector. However, although AI offers amazing forecasting tools, it is easy to misuse them, as the complexity and large amount of data can lead to models that appear accurate but fail under real-world conditions. The use of AI models does not inherently exclude the pitfalls of overfitting.

In institutional settings, there is also a high level of skepticism about using AI innovative tools for the following reasons:

- If AI models perfectly replicated the behavior of human traders, considering that 95% of human traders lose money in financial markets, 95% of AI models would also be destined to lose money in the markets
- The mass adoption of AI models in financial markets could change the market structure in terms of prices and volumes, rendering the historical datasets on which the models are trained obsolete
- Major market players are investing significant capital into development teams and big data maintenance, leading to a struggle among major players and excluding smaller ones from the market

Individual investors and small funds should consider investing in hybrid models that combine statistical know-how from the data science world with traditional approaches based on technical analysis and algorithmic trading. Their goal should be to create personalized, low-cost AI models where the edge lies not only in the technology itself but also in their own individual expertise, which can be greatly enhanced by AI.

This chapter, like the entire book, is geared toward traders. It focuses on using available tools, including those developed by others, to analyze and trade markets effectively. Among these tools, AI-based solutions are particularly powerful. The chapter will explore how to integrate EasyLanguage with Python AI models, merging traditional trading techniques with advanced analytics to equip traders with robust decision-support tools.

In this chapter, we are going to cover the following:

- The main difference between Python and EasyLanguage
- Volatility Predictor on gold futures by the Gandalf Project
- Bridging Python and EasyLanguage
- Embedding AI predictions into EasyLanguage indicators
- Embedding AI predictions into EasyLanguage strategies
- Creating a volatility dashboard for multiple assets

- Using TradeStation to provide prediction strength from multiple AI models

Python versus EasyLanguage

Python is a versatile, interpreted programming language valued for its clear and straightforward syntax, which makes it accessible to both novice and seasoned developers. It accommodates various programming styles, such as procedural, object-oriented, and functional programming. With a rich standard library and a vast ecosystem of third-party modules, Python is well-suited for a wide range of tasks, from web development and data analysis to automation and beyond.

Python's broad applicability contrasts with EasyLanguage's specialized, trader-friendly focus, data storage, real-time execution, technical trading libraries, and easily customizable RadarScreen and trading dashboard.

EasyLanguage works exclusively within the TradeStation platform, so the idea to bridge Python and TradeStation arises from the finance community's need to leverage complex AI model outputs without requiring in-depth knowledge of data science. This integration allows traders and financial analysts to utilize sophisticated machine learning and AI models developed in Python while continuing to operate within the familiar and specialized environment of TradeStation. By combining Python's advanced analytical capabilities with TradeStation's robust trading features, users can enhance their trading strategies with cutting-edge AI insights.

Let's see how to do that. In *Figure 10.1*, you can see a basic pipeline to explain the bridging process:

Figure 10.1 – Bridging Python and TradeStation

In *Figure 10.1*, you can see that Python, after going through all the steps to develop AI applications, can be programmed to export predictions into EasyLanguage functions to be embedded into indicators, strategies, and RadarScreen. The first thing we need to start is, of course, an AI model.

Volatility Predictor on gold futures by the Gandalf Project

Volatility Predictor is an AI-based model designed to provide volatility predictions for the coming day. For the purpose of this chapter, we are going to use a Python-based model developed by the Gandalf Project (https://gandalfproject.com). This model exploits the powerful **Extreme Gradient Boosting** (**XGBoost**) Python library to predict the coming day's volatility; as you will see, any Python model can be programmed to be bridged to EasyLanguage functions.

Volatility Predictor's purpose is to provide a volatility prediction class for the next day ranging from `0` to `4`, according to *Table 10.1*:

Volatility Predictor class	Description
`0`	Expected volatility between the 0th and 20th percentile
`1`	Expected volatility between the 20th and 40th percentile
`2`	Expected volatility between the 40th and 60th percentile
`3`	Expected volatility between the 60th and 80th percentile
`4`	Expected volatility between the 80th and 100th percentile

Table 10.1 – Volatility Predictor classes

In *Table 10.1*, you can see the Volatility Predictor classes legend, whereby classes range from the lowest volatility class (with volatility expected between the 0th and 20th percentiles) to the highest volatility class, in the 80th to 100th percentile range.

> ***NOTE***
>
> *A percentile is a statistical measure that indicates the value below which a given percentage of observations in a group of observations falls. For example, the 25th percentile is the value below which 25% of the observations may be found.*

Bridging Python and EasyLanguage

To bridge Python and EasyLanguage, we must first add (or have the AI model developers add) these lines of Python code:

```
#export TS function
from datetime import datetime
dates = pd.to_datetime(dataset.index, format='%Y%m%d')
dataset['datestr'] = dates.strftime('%Y-%m-%d')
dataset['yy']      = dataset['datestr'].str[0:4].astype(int)
```

```
dataset['yy_']     = dataset['yy']-1900
dataset['yy__']    = dataset['yy_'].apply(str)
dataset['dd']      = dataset['datestr'].str[-2:]
dataset['mm']      = dataset['datestr'].str[5:7]
dataset['Tsdate'] = dataset['yy__']+dataset['mm']+dataset['dd']
dataset['Tsdate']
dataset['code']='if date='+dataset['Tsdate'].astype(str)+' then
volatility_prediction='+dataset['y_pred'].astype(str)+';'
dataset['code'].to_csv('volatility_predictions.txt',index=False)
dataset['code']
```

The preceding Python code block will produce the **`volatility_predictions.txt`** file, which will look as shown in *Figure 10.2*:

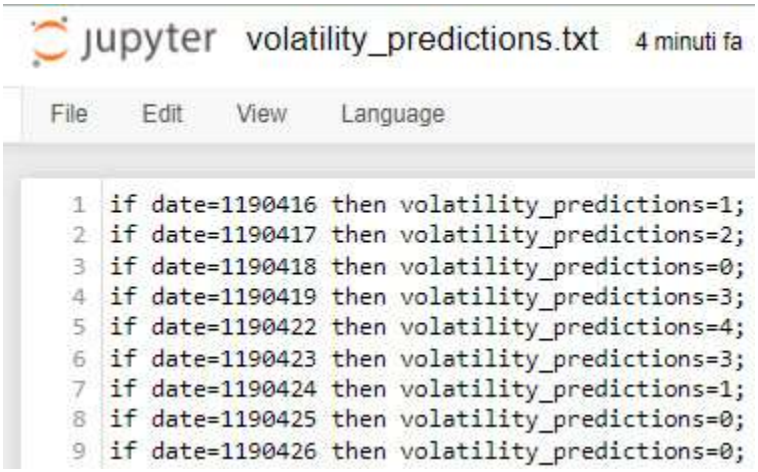


Figure 10.2 – The volatility_predictions.txt file

In *Figure 10.2*, you can see what the **`volatility_predictions.txt`** file looks like. It provides the EasyLanguage code that you can copy and paste into the EasyLanguage **`Volatility_Predictions`** function (*Figure 10.3*):

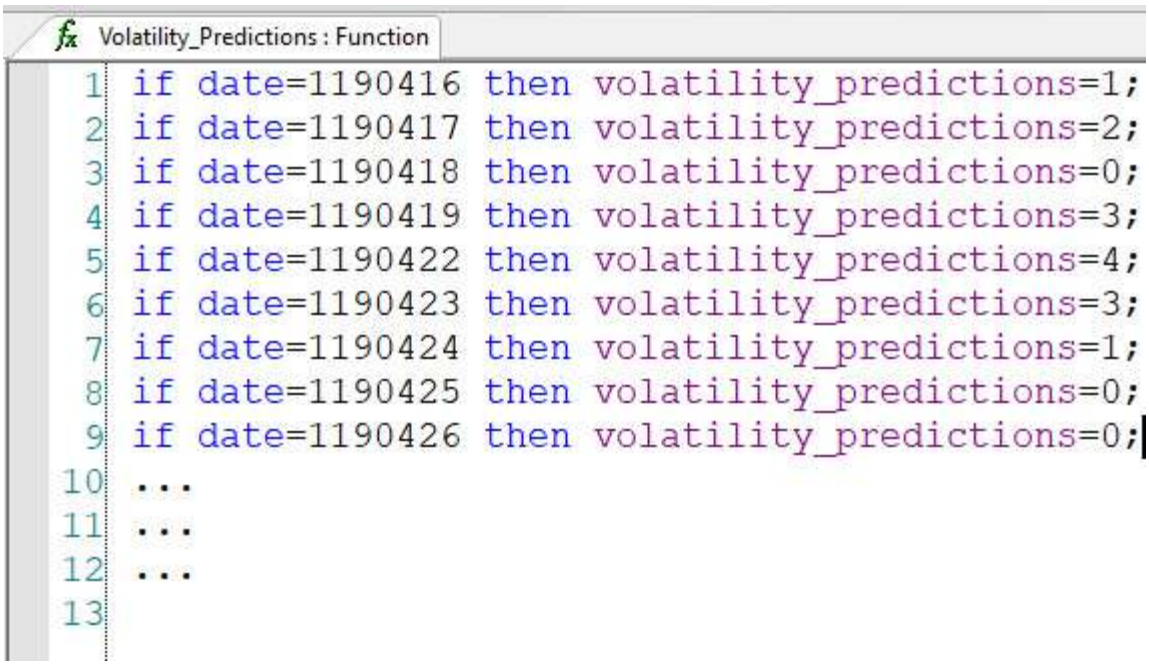


Figure 10.3 – The EasyLanguage Volatility_Predictions function

As you can see in *Figure 10.3*, the EasyLanguage function assigns a **`Volatility`** class for each day. For example, in row 1, it can be seen that the volatility forecast class for April 16th, 2019 was **`1`**. In other words, the trader, at the close of that day, was alerted that the predicted volatility for the next day fell into the 20th–40th percentile range, indicating that a low volatile day was expected.

Looking at row 5, you can see that at the close of April 22nd, 2019, there was a volatility prediction of **`4`**, meaning a volatility falling in the 80th–100th percentile range was excepted.

You can find more information on this model in the *How To Enhance A Trading Strategy On Gold Futures By Using An AI-Based Volatility Predictor* article, published in the July 2024 *Stocks & Commodities* issue available at https://traders.com/#Jul2024.

The trading rationale behind a volatility prediction is to filter breakout trades only if the next day's volatility is expected to be high and to stay out of the market if the volatility expectation is low. As you will remember from *Chapter 8*, an intraday breakout strategy holds positions for a few hours, so filtering entries only to catch expected big price movements could enhance trading performances.

In *Figure 10.4*, you can see the Volatility Predictor pipeline:

Figure 10.4 – Volatility Predictor pipeline

From *Figure 10.4*, we can see that by programming an AI Python model, we can create EasyLanguage functions that provide volatility predictions for the coming day. We can then program EasyLanguage to embed these functions into indicators and strategies.

Now, let's see how to embed the Python outputs into EasyLanguage.

Embedding AI predictions into EasyLanguage indicators

To embed the **`Volatility_Predictions`** function into an EasyLanguage indicator, we can write the following:

```
var:color(0);
if Volatility_Predictions=0 then color=yellow;
if Volatility_Predictions=1 then color=red;
if Volatility_Predictions=2 then color=darkred;
if Volatility_Predictions=3 then color=darkgreen;
if Volatility_Predictions=4 then color=green;
plot1(Volatility_Predictions,"Volty_pred",color)
```

The **`Volatility_Predictions`** command calls the EasyLanguage function created by importing the Python **`.txt`** file. The **`plot1(Volatility_Predictions,"Volty_pred",color)`** command will plot the **`volatility predictions`** class in a TradeStation subchart, giving each class a different color, as in *Figure 10.5*:

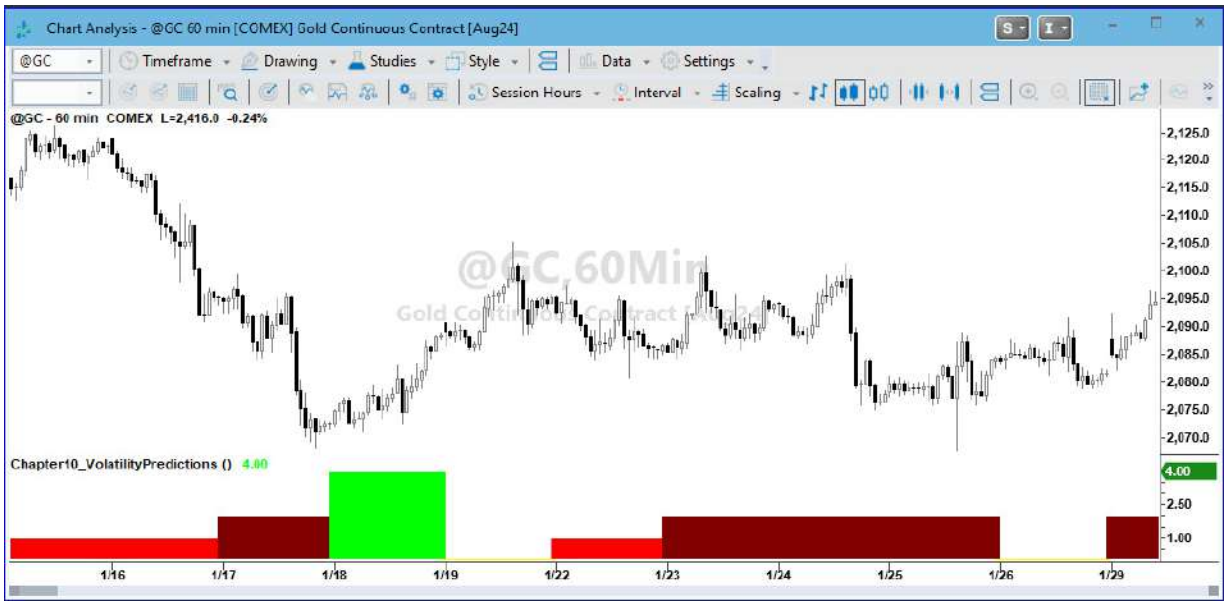


Figure 10.5 – Volatility_Predictions indicator

In *Figure 10.5*, you can see the volatility predictions class day by day on a 60-minute gold futures chart. As we learned in the previous chapters, once we have visual information on a chart, we can move further to program strategies.

Embedding AI predictions into EasyLanguage strategies

Let's now move on to how to embed the **`Volatility_Predictions`** function into an EasyLanguage strategy. In this section, we are going to learn how to use the Volatility Predictor model for the following:

- Filtering trades
- Money management
- Position management

Using the Volatility Predictor model as a filter

To use the Volatility Predictor to filter trades, let's start with the basic Intraday Breakout Strategy we learned in *Chapter 8*:

```
input:SR_Time(800),SR_BarsBack(12),WaitBars(1);
var:Resistance(0),Support(0),SR_count(0);
if time=SR_Time then Begin
    Resistance=highest(H,SR_BarsBack);
    Support=lowest(L,SR_BarsBack);
    SR_count=0;
End;
```

```
SR_count=SR_count+1;
if time>=SR_Time and SR_count<Waitbars then begin
    buy       next bar Resistance stop;
    sellshort next bar Support stop;
End;
Input:BarsIN(24);
if BarsIN>0 and barssinceentry = BarsIN then Begin
    if mp = 1 then sell next bar open;
    if mp =-1 then buytocover next bar open;
End;
```

Starting from this script, we can proceed to embed the **`Volatility_Predictions`** function from the Python model, as in the following script:

```
Input:VolatilityFilter(0);
input:SR_Time(800),SR_BarsBack(12),WaitBars(1);
var:Resistance(0),Support(0),SR_count(0);
if time=SR_Time then Begin
    Resistance=highest(H,SR_BarsBack);
    Support=lowest(L,SR_BarsBack);
    SR_count=0;
End;
SR_count=SR_count+1;
Volatility_Predictions=VolatilityFilter And
    time>=SR_Time and SR_count<Waitbars then begin
        buy       next bar Resistance stop;
        sellshort next bar Support stop;
End;
Input:BarsIN(24);
if BarsIN>0 and barssinceentry = BarsIN then Begin
    if mp = 1 then sell next bar open;
    if mp =-1 then buytocover next bar open;
End;
```

In the previous script, we added a new input, **`Input:VolatilityFilter(0)`**, as well as the **`VolatilityPredictions=VolatilityFilter`** filter.

The combination of these two modifications allows us to filter trades based on the daily volatility forecasted class provided by the AI model.

In *Figure 10.6*, you can see the Intraday Breakout Strategy signals filtered by the AI-based volatility prediction with class **`4`**.

Figure 10.6 – Using Volatility Predictor to filter trades

In *Figure 10.6*, you can see how the strategy executes trade only if the volatility for the coming day is expected to be in the 80th–100th percentile range, thus integrating a simple technical signal (the intraday breakout) with a forecast from a much more complex AI model.

Once the volatility predictions are integrated into the EasyLanguage code, we can utilize all the basic TradeStation functions learned in the previous chapters. For example, we can run simulations to determine what type of volatility filter to apply. In *Figure 10.7*, you can see the optimization results obtained by varying the **`VolatilityFilter`** input from **`0`** to **`4`** in the last three years' historical period:

	Chapter10 Futures Breakout: VolatilityFilter	All: Net Profit	All: Total Trades	All: Gross Profit	All: % Profitable	All: Return on Account
1	4	31,400.00	197	144,410.00	43.65	113.52
2	3	13,690.00	188	119,950.00	42.02	64.27
3	2	-27,030.00	217	93,030.00	39.17	-85.21
4	1	-23,440.00	140	54,680.00	37.86	-70.79
5	0	15,210.00	181	109,230.00	39.23	56.86

Figure 10.7 – Volatility Predictor for filtering trades: optimization results 1

From *Figure 10.7*, we can see that filtering trades with **`VolatilityFilter=4`**, meaning filtering trades only if the volatility expected is in the 80th–100th percentile range, definitely improves the **`Net Profit`**, **`% Profitable`**, and **`Return on Account`** values.

Another interesting way to test the **`VolatilityFilter`** is to replace the **`if Volatility_Predictions=VolatilityFilter`** condition with the **`if Volatility_Predictions>=VolatilityFilter`** condition.

By applying this change, we can see how the strategy works from a broader perspective. With this small modification, we can answer questions such as *what happens if we filter trades with volatility prediction classes from 3 to 4, meaning within the 60th–100th percentile range?*, *what happens if we exclude from trading only the 0 class?*, and so on. In *Figure 10.8*, you can see the optimization results:

	Chapter10 Futures Breakout: VolatilityFilter	All: Net Profit	All: Total Trades	All: Gross Profit	All: % Profitable	All: Return on Account
1	4	46,430.00	302	230,110.00	45.36	167.86
2	3	66,350.00	472	339,360.00	44.92	261.53
3	2	54,710.00	660	418,220.00	44.09	167.05
4	1	29,230.00	774	448,480.00	42.25	82.50
5	0	18,090.00	896	508,230.00	41.29	34.98

Figure 10.8 – Volatility Predictor for filtering trades: optimization results 2

In *Figure 10.8*, you can see the result of backtesting according to the amended code and draw the following conclusions:

- As we increase the volatility filter, the number of trades decreases, dropping from **`896`** with **`VolatilityFilter>=0`** (which means no volatility filter applied) to **`302`** trades with **`VolatilityFilter>=4`** (which means filtering trades only if volatility is expected to be in the 80th to 100th percentile range)
- As we increase the volatility filter, the **`Net Profit`** value increases from 18,090 USD with **`VolatilityFilter>=0`**, then shows a peak at 66,350 USD with **`VolatilityFilter>=3`**, then drops to 46,430 USD with **`VolatilityFilter>=4`**
- As we increase the volatility filter, the **`% Profitable`** value increases from 41.29% with **`VolatilityFilter>=0`** to 45.36% with **`VolatilityFilter>=4`**
- As we increase the volatility filter, the **`Return on Account`** value increases from **`34.98`** with **`VolatilityFilter>=0`**, then shows a peak at **`261.53`** with **`VolatilityFilter>=3`**, then drops to **`167.86`** with **`VolatilityFilter>=4`**

We can conclude this analysis by saying that filtering trades according to the volatility predictions provided by the AI Volatility Predictor model generally improves the main performance indicator, showing the best volatility filter at **`VolatilityFilter>=3`**.

Using the Volatility Predictor model for money management

If the expected volatility is high, it makes sense to widen stop-loss and target levels. Conversely, if the expected volatility is low, we might decide not to trade or to use closer stop and target levels. To use volatility forecasts for money management purposes, we can write the following:

```
input:SR_Time(800),SR_BarsBack(12),WaitBars(1);
```

```
var:Resistance(0),Support(0),SR_count(0);
if time=SR_Time then Begin
    Resistance=highest(H,SR_BarsBack);
    Support=lowest(L,SR_BarsBack);
    SR_count=0;
End;
SR_count=SR_count+1;
if time>=SR_Time and SR_count<Waitbars then begin
        buy       next bar Resistance stop;
        sellshort next bar Support stop;
End;
Input:AtrsHighVolty(4),AtrsLowVolty(1);
var: SlossLevel(0),TgtLevel(0);
var:ATR(0);
ATR = Avgtruerange(20);
var:NumAtr(0);
if Chapter10_VolatilityPredictions<3  then NumAtr=AtrsLowVolty;
if Chapter10_VolatilityPredictions>=3 then NumAtr=AtrsHighVolty;
if mp =1 then Begin
    sell ("slL") next bar entryprice - NumAtr*ATR stop;
    sell ("tgtL") next bar entryprice + NumAtr*ATR limit;
End;
if mp =-1 then Begin
    buytocover ("slS") next bar entryprice + NumAtr*ATR stop;
    buytocover ("tgtS") next bar entryprice - NumAtr*ATR limit;
End;
```

In the preceding code, we added two new inputs. The **`AtrsHighVolty`** input will widen the stop loss and target distance when the volatility expected is high, and the **`AtrsLowVolty`** input will narrow the stop loss and target distance when the volatility expected is low.

In *Figure 10.9*, you can see the optimization results obtained by varying the two inputs from **`1`** to **`4`**:

	Chapter10 Volatility Predictor for Money Management: AtrsHighVolty	Chapter10 Volatility Predictor for Money Management: AtrsLowVolty	All: Net Profit	All: % Profitable	All: Return on Account	All: Avg Trade
1	2	1	31,660.00	48.81	179.17	16.00
2	3	1	34,660.00	47.62	124.23	19.21
3	3	2	26,840.00	46.32	84.88	19.16
4	4	1	19,140.00	46.74	68.16	11.04
5	3	4	23,010.00	43.08	67.20	20.82
6	2	4	17,520.00	46.39	59.25	13.46
7	2	2	15,360.00	47.86	53.89	9.67
8	3	3	19,320.00	43.35	52.77	16.26
9	2	3	11,990.00	46.08	43.22	8.70
10	4	2	7,030.00	44.78	20.96	5.28
11	4	4	6,020.00	41.91	15.94	5.87
12	4	3	4,430.00	42.00	11.84	3.98
13	1	1	-1,480.00	48.26	-6.69	-0.63
14	1	4	-12,670.00	46.62	-36.83	-7.44
15	1	3	-12,620.00	46.57	-37.65	-7.11
16	1	2	-17,410.00	47.57	-50.73	-8.81

Figure 10.9 – Optimization report by varying AtrsHighVolty and AtrsLowVolty from 1 to 4

As you see in *Figure 10.9*, the best **`Return on Account`** results are shown with **`AtrsHighVolty`** higher than **`AtrsLowVolty`**, demonstrating that it makes sense to widen stop loss and target when a high volatility trading session is expected.

Using the Volatility Predictor model for position sizing

We know from what we have learned in previous chapters that an intraday breakout strategy performs better on high-volatility days. Therefore, it seems like a good idea to test a position sizing technique that uses a larger size when the expected volatility is high and a smaller size when the expected volatility is low. To test this idea, we can write the following:

```
input:SR_Time(800),SR_BarsBack(12),WaitBars(1);
var:Resistance(0),Support(0),SR_count(0);
if time=SR_Time then Begin
    Resistance=highest(H,SR_BarsBack);
    Support=lowest(L,SR_BarsBack);
    SR_count=0;
End;
SR_count=SR_count+1;
Input:QtyForHighVolty(2),QtyForLowVolty(1);
var:Qty(0);
if Volatility_Predictions>=3 then Qty = QtyForHighVolty;
if Volatility_Predictions<3  then Qty = QtyForLowVolty;
if time>=SR_Time and SR_count<Waitbars then begin
        buy       next bar Qty shares Resistance stop;
        sellshort next bar Qty shares Support stop;
End;
Input:BarsIN(24);
if BarsIN>0 and barssinceentry = BarsIN then Begin
    if mp = 1 then sell next bar open;
    if mp =-1 then buytocover next bar open;
End;
```

In the preceding code, we added the two inputs, **`QtyForHighVolty`** and **`QtyForLowVolty`**, with the aim to increase sizes when the expected volatility is high and to reduce them when the expected volatility is low.

In *Figure 10.10*, you can see an optimization report obtained by varying the two new inputs from **`1`** to **`3`**:

	Chapter10 Volatility Predictor for Position Sizing: QtyForHighVolty	Chapter10 Volatility Predictor for Position Sizing: QtyForLowVolty	All: Net Profit	All: % Profitable	All: Return on Account	All: Avg Trade
1	3	1	78,570.00	40.10	75.31	100.99
2	2	1	36,990.00	40.10	47.41	47.54
3	3	2	32,400.00	40.10	24.97	41.65
4	1	1	-4,590.00	40.10	-8.88	-5.90
5	2	2	-9,180.00	40.10	-8.88	-11.80
6	3	3	-13,770.00	40.10	-8.88	-17.70
7	1	2	-50,760.00	40.10	-49.58	-65.24
8	2	3	-55,350.00	40.10	-37.07	-71.14
9	1	3	-96,930.00	40.10	-60.93	-124.59

Figure 10.10 – Optimization of the Volatility Predictor model for position sizing

In *Figure 10.10*, you can see the optimization report obtained by varying **`QtyForHighVolty`** and **`QtyForLowVolty`** from **`1`** to **`3`**. As you see, the best **`Return on Account`** results are shown when **`QtyForHighVolty`** is higher than **`QtyForLowVolty`**, demonstrating that it makes sense to reduce size when the expected volatility is high and reduce it when the expected volatility is low.

> ***NOTE***
>
> *In financial theory, to reduce risk, portfolio sizes are generally decreased as volatility increases, particularly in long equity market strategies. However, in long/short intraday breakout strategies, the opposite is true: in anticipation of low volatility, it's better to stay out of the market or reduce exposure.*

Creating a volatility dashboard for multiple assets

Key financial assets such as gold, oil, major currencies, and stock markets are influenced by the same economic and geopolitical factors. As a result, significant price movements often occur simultaneously across different assets, sometimes moving in the same direction and sometimes in opposite directions. This is why a trading desk must have a dashboard that provides an overall view. The Volatility Predictor model can be used for this purpose.

If we apply the same AI model to other financial instruments, we can gain a useful global view of the market through TradeStation's RadarScreen. Here's how.

Python models can be programmed to print EasyLanguage code for specific assets, as shown in *Figure 10.11*:

```
if symbol="@GC" then begin
    if date = 1070102 then Multiple_Volatility_Predictions =4;
    if date = 1070103 then Multiple_Volatility_Predictions =3;
    if date = 1070104 then Multiple_Volatility_Predictions =2;
    if date = 1070105 then Multiple_Volatility_Predictions =3;
    ...
End;
if symbol="@CL" then begin
    if date = 1070102 then Multiple_Volatility_Predictions =0;
    if date = 1070103 then Multiple_Volatility_Predictions =3;
    if date = 1070104 then Multiple_Volatility_Predictions =2;
    if date = 1070105 then Multiple_Volatility_Predictions =1;
    ...
End;
if symbol="@TY" then begin
    if date = 1070102 then Multiple_Volatility_Predictions =3;
    if date = 1070103 then Multiple_Volatility_Predictions =4;
    if date = 1070104 then Multiple_Volatility_Predictions =0;
    if date = 1070105 then Multiple_Volatility_Predictions =4;
    ...
```

Figure 10.11 – The Multiple_Volatility_Predictions function

In *Figure 10.11*, you can see an example of an EasyLanguage function for multiple assets. By using this function on a TradeStation RadarScreen, you will obtain an output as shown in *Figure 10.12*.

Symbol	Interval	Chapter10_VolatilityPredictions_RadarScreen			
		Volty_Class 3days ago	Volty_Class 2days ago	Volty_Class 1day ago	Volty_Class today
@GC	Daily	4	3	2	1
@CL	Daily	1	2	3	4
@TY	Daily	3	4	2	0
@ES	Daily	3	3	2	0

Figure 10.12 – TradeStation RadarScreen with volatility predictions on multiple assets

In *Figure 10.12*, you can see how RadarScreen can be used to provide the volatility prediction at a glance for the coming day. RadarScreen shows four columns representing, from left to right, the last four sessions. Such a layout helps track the short-term model's accuracy.

Using TradeStation to collect predictions from multiple AI models

I have met dozens of excellent data scientists and worked with them in teams. The main difficulty I encountered, which I can understand, is that they are often very protective of their know-how and reluctantly share it with their colleagues. So, I found it very useful to allow each of them to work independently while following common guidelines, such as the number of volatility classes and the `.txt` output of the model. I then used TradeStation to gather and process the forecasts from the different models to calculate a prediction score according to *Figure 10.13*:

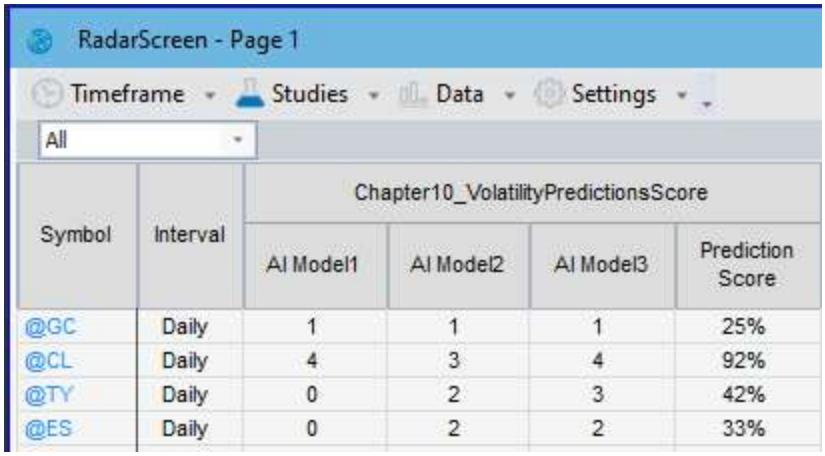
Figure 10.13 – Pipeline to calculate prediction score

Let's see how to perform the process described in *Figure 10.13* by looking at the following EasyLanguage code:

```
var:Score(0);
value1=Multiple_Volatility_Predictions;
value2=Multiple_Volatility_Predictions1;
value3=Multiple_Volatility_Predictions2;
score = ((value1+value2+value3)/3)/4;
plot1(value1,"AI Model1");
plot2(value2,"AI Model2");
plot3(value3,"AI Model3");
plot4(score,"Score");
```

The preceding code collects predictions coming from three different Python AI models and then calculates the score (ranging from 0% to 100%) by averaging the three predictions and dividing the result by `4`, which is the highest possible reading. The higher the score, the higher the expected volatility.

In *Figure 10.14*, you can see what RadarScreen looks like:



Symbol	Interval	Chapter10_VolatilityPredictionsScore			
		AI Model1	AI Model2	AI Model3	Prediction Score
@GC	Daily	1	1	1	25%
@CL	Daily	4	3	4	92%
@TY	Daily	0	2	3	42%
@ES	Daily	0	2	2	33%

Figure 10.14 – RadarScreen showing the prediction score

In *Figure 10.14*, you can see RadarScreen showing the prediction provided by three different AI models along with the prediction score. For example, you can see that for crude oil (`@CL`), the three models converge to predict a high volatility session for the coming day with a score of 92%.

This approach not only addresses privacy concerns, as each data scientist shares only their outputs rather than the entire model, but it also enhances the robustness of the decision-making model.

Summary

As you can imagine, the trader's goal is not programming but surviving (in the early stages of their career) and then confidently navigating the financial markets. A trader must also evaluate the vast array of technology and trading tools available on the market, understanding that software, whether it's Python or EasyLanguage, is merely a tool (think of it like a calculator). Success does not come from the tool itself but from the scientific logic applied and your motivation. In the institutional world, there is a saying, "*Garbage in, garbage out*," meaning that if you use a sophisticated model lacking sound underlying logic, you're simply producing "amazing" garbage.

Traders lacking programming skills for AI tools should at least understand the dynamics behind them and delegate the task of producing AI models under their supervision to data scientists.

In this chapter, we learned how to integrate Python AI models with EasyLanguage functions, as well as how to use TradeStation features embedding the AI model predictions into indicators, strategies, and RadarScreen. Then, we learned how to use TradeStation to collect predictions from several AI models and produce a prediction score RadarScreen, thus obtaining a more robust decision support system enhanced by AI.

In summary, EasyLanguage is a programming language that operates exclusively within the TradeStation platform, while Python is a versatile, general-purpose programming language. Python's flexibility allows it to be used to *write functions in EasyLanguage*, enabling the integration and utilization of potentially *any* AI models, from machine learning to natural language processing, within TradeStation.

In the next chapter, we will delve deeper into one of the most fascinating areas of AI applied to financial markets: machine learning for pattern recognition.

11

EasyLanguage for Machine Learning

Artificial intelligence (**AI**), when applied to financial markets, is a vast field. However, in my experience, while AI is fascinating from a programming perspective, not all AI tools are suitable for guaranteeing financial performance in the real market, with many of them more theoretical than practical.

This chapter aims to provide you with tools that, based on my experience, can be used effectively in market trading; any theoretical discussion about AI applied to financial markets is beyond the scope of this book.

In this chapter, we'll explore a specific area within AI tools – machine learning for pattern recognition. We will then demonstrate how EasyLanguage, although not as advanced as Python, can serve as a valuable tool for traders to integrate advanced learning logic into TradeStation's suite, without the need for specialized data science competencies.

To this end, we'll work in parallel with one of the early pattern recognition experiments conducted by Sir R. Fisher, a mathematician and biologist, in the 1930s.

In this chapter, we're going to cover the following topics:

- A definition of machine learning for pattern recognition
- The Iris dataset and Fisher's project
- Labeling trading sessions
- Project N.1 – recognizing up sessions
- The confusion matrix in EasyLanguage
- Project N.2 – recognizing down sessions

A definition of machine learning for pattern recognition

Machine learning is a broad field within AI that focuses on developing algorithms that allow computers to learn from data and make decisions or predictions, without being explicitly programmed for specific tasks. The primary goal is to enable machines to improve their performance on a given task through experience.

Pattern recognition is a specific application area within the broader field of machine learning. It involves the process of identifying patterns and regularities in data. Pattern recognition is essentially about categorizing or classifying data based on learned patterns.

In my experience, I've found that there are numerous similarities between the logic behind machine learning for pattern recognition and those of the traditional technical trading approach.

Every time traders analyze a stock chart, they search, sometimes unconsciously, for a price configuration that they have seen before. If you listen to experienced traders while discussing technical analysis, you will hear phrases such as, "*As the price has behaved this way up until now, it might behave in this way in, say, the next couple of days*" or "*To reach this higher resistance level, the price should first drop to this support level and then bounce back.*"

An experienced trader opens a new position because they recognize something on a chart that they have seen in the past and, most likely, know how it will play out based on previous outcomes.

As you can see, there are many similarities between machine learning for pattern recognition and the traditional trading approach; this is why I believe this area is, among the many alternatives that AI provides, the most exciting for trade financial markets. Let me tell you how this adventure began for me.

The Iris dataset and Fisher's project

While studying Python and its numerous libraries for machine learning, I came across the Iris dataset and the pattern recognition project that Sir R. A. Fisher, a scientist from the 1930s, undertook in the field of biology. The more I delved into Fisher's project, the more I realized how many aspects of machine learning for pattern recognition align with trading. This intrigued me greatly, and I began to delve deeper into the subject on my own.

The Iris dataset (*Figure 11.1*) is a classic example of machine learning. It contains 150 samples of Iris flowers, with three species – Iris setosa, Iris versicolor, and Iris virginica. Each sample has four features – sepal length, sepal width, petal length, and petal width – measured in centimeters. The dataset is used to demonstrate classification algorithms as each species can be predicted based on the flower's measurements. Fisher's experiment aimed to show how these features could distinguish between the three species, making the dataset a foundational resource in statistical pattern recognition:

Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
51	7	3.2	4.7	1.4	Iris-versicolor
52	6.4	3.2	4.5	1.5	Iris-versicolor
53	6.9	3.1	4.9	1.5	Iris-versicolor
148	6.5	3	5.2	2	Iris-virginica
149	6.2	3.4	5.4	2.3	Iris-virginica
150	5.9	3	5.1	1.8	Iris-virginica

Figure 11.1 – The Iris dataset

In *Figure 11.1*, we can see that the dataset is composed of two parts: the **features** (sepal length, sepal width, petal length, and petal width) and the **target** (the species).

But why did Fisher set up the data this way? Why did he choose those four features and reject others, and why did he identify only three Iris species?

Well, Fisher wasn't just a mathematician; he was also a biologist. He chose that dataset structure because he understood that only a relationship between the selected features and the species would be meaningful. So, he decided to record the length of the petals and not, for example, their weight or their color because he knew, as a biologist, that weight and color have no relevance in the classification of the species.

It was a bit like the biologist in him knew there was a relationship between the features and the target and asked the mathematician in him to measure that relationship, so the mathematician found a way to identify a function that connected the features to the target. In machine learning, this is referred to as a **supervised learning approach**. This is highly relevant in financial markets, where it's easy to get lost in relationships between variables that lack any logical foundation.

NOTE

Supervised *and* ***unsupervised*** *are two primary types of machine learning, each with distinct characteristics and use cases. Supervised learning uses labeled data, while unsupervised learning uses unlabeled data. Supervised learning focuses on predicting outcomes, whereas unsupervised learning focuses on discovering hidden patterns or structures in data. Supervised learning performance can be easily measured using the ratio of the number of correct predictions made by the model to the total number of predictions (accuracy) since the true labels are known. Unsupervised learning performance is harder to measure and often relies on subjective evaluation or clustering validation indices. Most of the experienced traders I've spoken with prefer a supervised approach to machine learning, but this is because they aren't skilled enough to appreciate the unsupervised approach, which remains a possible and valid research path.*

Labeling trading sessions

Following in Fisher's footsteps, let's try to classify trading sessions. One of the most effective classifications, in my opinion, is based on comparing the highs and lows of the current trading session with those of the previous session. *Figure 11.2* shows four types of trading sessions:

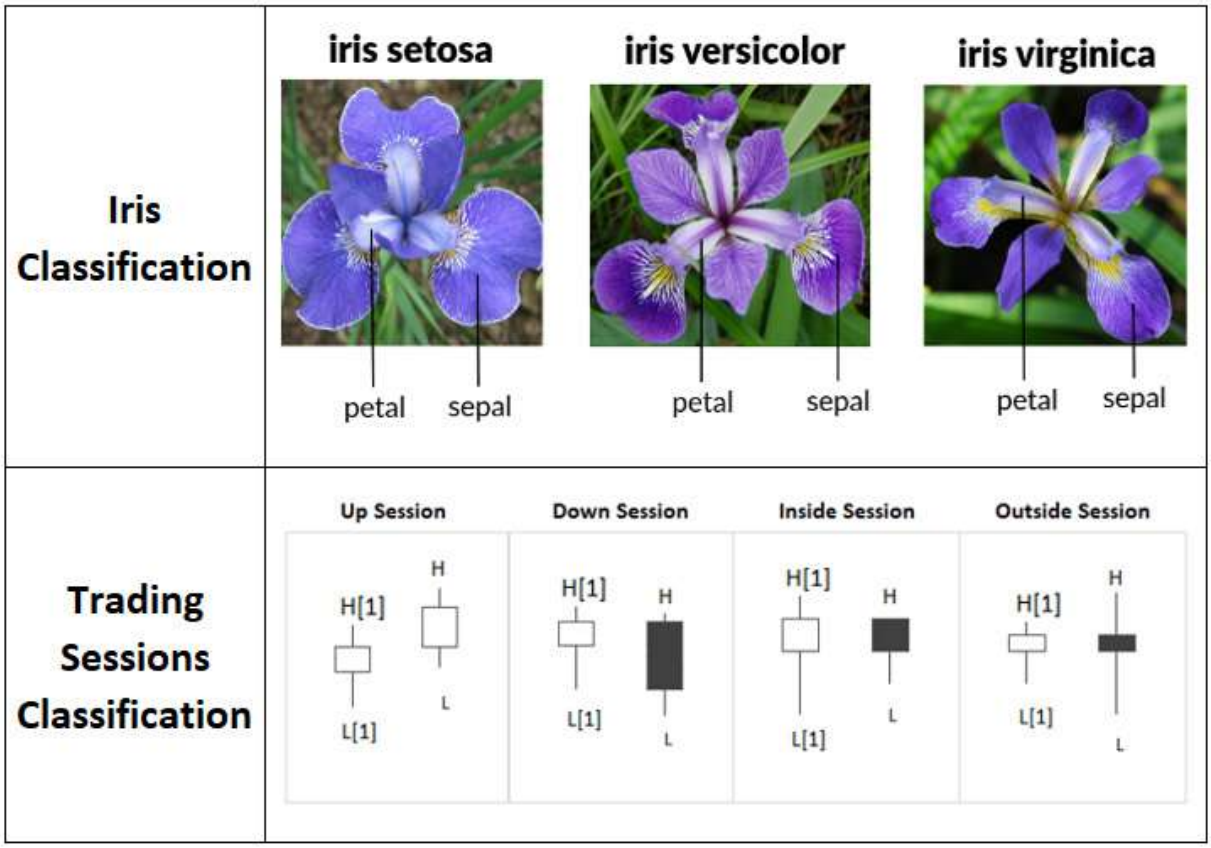


Figure 11.2 – Iris labels versus trading session labels

In *Figure 11.2*, you can see a possible trading session classification based on the relationship between the highs and lows of two consecutive trading sessions. As you can see, Fisher identified three species to recognize, whereas we have identified four trading session classes. The logic behind them is the same – labeling.

Table 11.1 provides more details on the trading session classification:

Trading Session	**What It Means**	**EasyLanguage Formula**
Up session	This is a bullish session showing both a higher high and a higher low	H>H[1] and L>L[1]

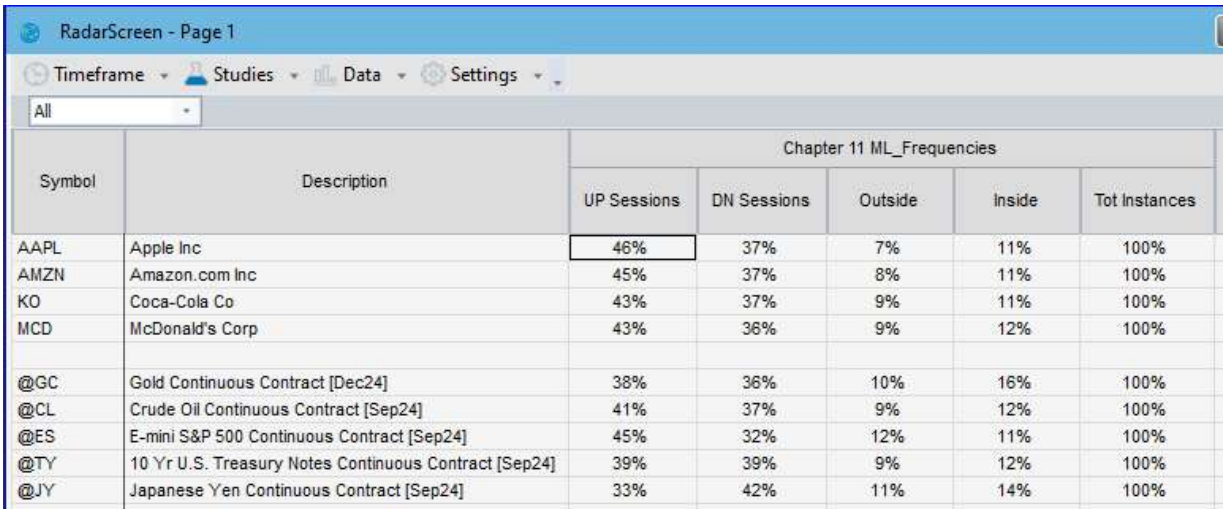
Trading Session	**What It Means**	**EasyLanguage Formula**
Down session	This is a bearish session showing both a lower high and a lower low	H<H[1] and L<L[1]
Inside session	This pattern typically indicates a period of consolidation or indecision in the market as buyers and sellers are relatively balanced	H<H[1] and L>L[1]
Outside session	The outside pattern is often seen as a sign of increased volatility and can indicate a strong shift in market sentiment, which can be a signal of potential trend reversal or continuation, depending on the context and subsequent price action	H>H[1] and L<L[1]

Table 11.1 – Trading session classification

Just as with the classification of Iris species, this method of classification encompasses all possible cases.

Before diving into the details of the model, the first question I like to ask when faced with a pattern to recognize is, how often does this pattern occur within the sample I'm analyzing? This step is crucial because it gives me an idea of the frequency with which the pattern appears in the dataset. Understanding this initial prevalence helps set realistic expectations for a model's performance and informs the choice of the most appropriate evaluation metrics. Without this insight, there's a risk of working with a pattern that is either too rare or too common, which could negatively impact the model's predictive effectiveness.

Figure 11.3 shows a TradeStation RadarScreen representing the four patterns' frequencies:



RadarScreen - Page 1

Timeframe Studies Data Settings

All

Symbol	Description	Chapter 11 ML_Frequencies				
		UP Sessions	DN Sessions	Outside	Inside	Tot Instances
AAPL	Apple Inc	46%	37%	7%	11%	100%
AMZN	Amazon.com Inc	45%	37%	8%	11%	100%
KO	Coca-Cola Co	43%	37%	9%	11%	100%
MCD	McDonald's Corp	43%	36%	9%	12%	100%
@GC	Gold Continuous Contract [Dec24]	38%	36%	10%	16%	100%
@CL	Crude Oil Continuous Contract [Sep24]	41%	37%	9%	12%	100%
@ES	E-mini S&P 500 Continuous Contract [Sep24]	45%	32%	12%	11%	100%
@TY	10 Yr U.S. Treasury Notes Continuous Contract [Sep24]	39%	39%	9%	12%	100%
@JY	Japanese Yen Continuous Contract [Sep24]	33%	42%	11%	14%	100%

Figure 11.3 – The patterns' frequencies

In *Figure 11.3*, you can see the patterns' frequency for a few stocks and a list of futures calculated on 10 years of data. The frequency distribution of patterns tells us a great deal. First and foremost, we can see that there are two primary patterns – an up session and a down session, which together account for around 80% of the total observations. In addition, there are two secondary patterns – inside and outside. This suggests that recognizing the primary patterns could be a moderately challenging task, whereas identifying rarer patterns such as inside and outside could prove to be more difficult.

Additionally, we can see that the frequency of the up session pattern is consistently higher in the stocks and in the S&P 500 index (**@ES**), which is a stock index. This is due to the tendency of stock markets to rise over time, a phenomenon known as **long bias**. Conversely, the yen futures (**@JY**) appear to be influenced by short bias, where downward movements are more prevalent.

Such information can be useful when analyzing backtesting results. Understanding the prevalence of different patterns and biases in various markets helps us contextualize a model's performance and interpret its outcomes more effectively.

The EasyLanguage code to print out the pattern frequencies is as follows:

```
var:UpSession(0),DnSession(0),Outside(0),Inside(0),Tot(0);
If H>H[1] and L>L[1] then UpSession=UpSession+1;
if H<H[1] and L<L[1] then DnSession=DnSession+1;
if H>H[1] and L<L[1] then Outside=Outside+1;
if H<H[1] and L>L[1] then Inside=Inside+1;
tot=tot+1;
If Tot<>0 then begin
   plot1(UpSession/Tot,"UP Sessions");
   plot2(DnSession/Tot,"DN Sessions");
   plot3(Outside/Tot,"Outside");
   plot4(Inside/Tot,"Inside");
End;
```

The preceding code can be applied to any TradeStation RadarScreen while following the instructions we covered in the previous chapters.

> ***NOTE***
>
> *In machine learning terminology, the frequency distribution of the target is often referred to as* ***prevalence****. It is calculated using the confusion matrix with the formula* $Prevalence = \frac{TP + FN}{Total\ Instances}$ *which represents the proportion of actual positive instances in a dataset, indicating how frequently the positive class occurs. Essentially, it shows the baseline probability of encountering a positive instance. This metric is useful for understanding the distribution of classes in the dataset, especially in imbalanced datasets, where one class might be much more common than the other. We will return to this subject in The confusion matrix section of this chapter.*

Project N.1—recognizing an up session

Let's start our first machine learning project. Based on Fisher's experiment, we're going to do the following:

- Select a target
- Select the features
- Perform the machine learning pipeline

Selecting the target

The objective of Project N.1 is to automatically recognize an up session (*Figure 11.4*) for AAPL (Apple Inc.) using a machine learning process:

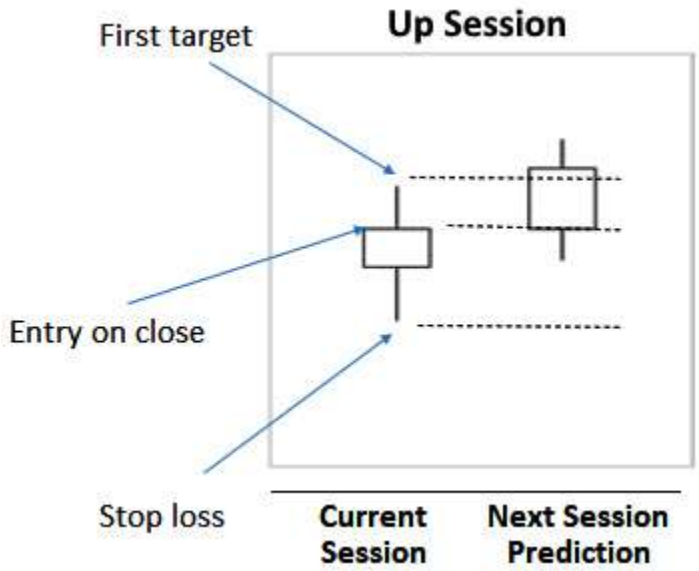


Figure 11.4 – An up session pattern

In *Figure 11.4*, you can see the target pattern for our first machine learning project. We chose this target for a reason; forecasting an **up session** allows a trader to plan a risk-calculated trade such as the following:

- Buying at the bar close
- Placing a stop loss under today's low
- Placing a first target on today's high
- Holding the position for one day

Let's look at a practical example. We'll assume that our model suggests a positive up session target for the following day. If the distance from today's high to the close is 1% and the distance from the close to the low is 0.4%, we can easily calculate the expected reward/risk ratio of our trade. The reward-to-risk ratio will be as follows:

$$RewRisk\ ratio = \frac{1\%}{0.4\%} = 2.5$$

So, the reward-to-risk ratio is 2.5, which is quite good, and we can consider placing the trade on close.

> ***NOTE***
>
> *In trading and machine learning, the term **target** has different meanings based on the context. In trading, target refers to a specific price level at which a trader aims to exit a trade to make a profit. In machine learning, **Target** refers to the output variable or the value that the model is trying to predict. To avoid confusion, we will use the term "Target" with an initial capital letter when referring to machine learning.*

Selecting the features

There are thousands of possible features that can be studied. However, just as Fisher did in his project, we will start with those that, based on experienced traders' wisdom, have a relationship with the target. This type of knowledge can also be found in technical analysis, a field that has been enriched for decades by traders around the world. Would it make sense to ignore it? For this first project, we will use two features – two **simple moving averages** (**SMAs**) and a **relative strength indicator** (**RSI**).

Table 11.2 shows what our dataset will look like:

ID (Date)	**Feature1** **Fast SMA > Slow SMA**	**Feature2** **RSI > Threshold**	**Target** **High and Low Higher than Previous Sessions**

1	0=false, 1=true	0=false, 1=true	0=false, 1=true
2	0=false, 1=true	0=false, 1=true	0=false, 1=true
3	0=false, 1=true	0=false, 1=true	0=false, 1=true

Table 11.2 – Our dataset

In *Table 11.2*, you can see the structure of our dataset, which consists of two features and one target:

- **Feature1**: This is a binary field represented by a FastSma above a SlowSma; it will be 1 if the FatsSma is above the SlowSma and 0 if it is below
- **Feature2**: This is a binary field represented by an RSI above a threshold; it will be 1 if the RSI is above the threshold and 0 if it is below
- **Target**: This is a binary field as well, represented by the price condition H>H[1] and L>L[1]; it will be 1 if today's high is higher than yesterday's high and today's low is above yesterday's low

Building the pipeline

In machine learning, a **pipeline** refers to a sequence of data processing and modeling steps arranged in a specific order, automating the workflow from raw data to a final model or prediction. This concept is central to creating efficient and reproducible machine learning models as it streamlines the process of transforming data, selecting features, training models, and making predictions.

Let's try to imagine how Fisher organized his research. Being a scientist, he likely planned the details first and then created some sort of procedure, essentially working out a pipeline. *Figure 11.5* shows the pipeline we're going to go through:

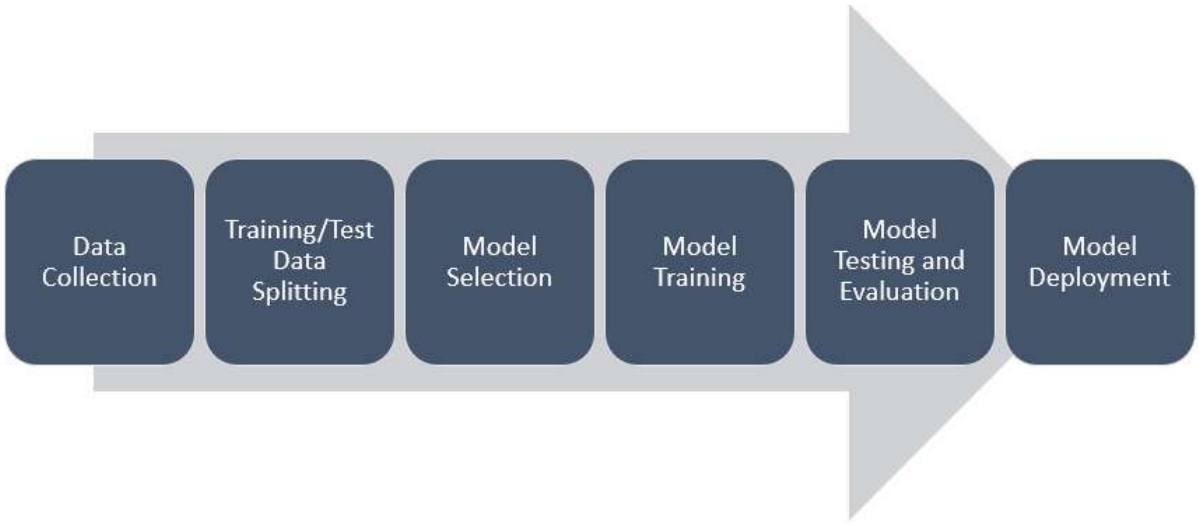


Figure 11.5 – A machine learning for pattern recognition pipeline

In *Figure 11.5*, you can see the machine learning pipeline for Project N.1; let's go through it step by step.

Data collection

Generally, a trader thinks of the data to analyze in the classic format – date, open, high, low, and close. However, in machine learning, data needs to be structured as in *Table 11.2*. To achieve this task, let's take a look at some basic EasyLanguage scripts to calculate the target and the features.

Note that our dataset is not a time series one; the fact that each row has an ID that is a date should not be misleading. The dataset could be randomly ordered and our machine learning project would still be valid. Traditional algorithmic trading platforms cannot handle datasets other than time series, which is a major limitation because it forces traders to analyze data only in a temporal sequence. In other words, this represents a real novelty for those working in financial markets – analyzing markets without relying on historical time series.

Target formula

The EasyLanguage code block to write the target is as follows:

```
var:Target_(0);
If H>H[1] and L>L[1] then Target_=1 else Target_=0;
if Target_=1 then Begin
    plot1(H,"H",green);
    plot2(L,"L",green);
End;
```

The **`If H>H[1] and L>L[1] then Target_=1 else Target_=0;`** command identifies a target any time the day shows both a high higher than yesterday's one and a low higher than yesterday's one.

Feature formulas

For the two features, as we learned in *Chapters 3* and *4*, we can write the following:

```
var:Feature1(0),Feature2(0);
if average(close,20)[1]>average(close,200)[1] then Feature1=1 else Feature1=0;
if RSI(Close,3)[1] > 50                       then Feature2=1 else Feature2=0;
```

The **`[1]`** command, related to the features, is essential as it ensures that the value of the features is calculated based on the session before the target.

To proceed along our pipeline, it is a good idea (as we learned in the early chapters of this book) to visualize the variables we're working on. In *Figure 11.6*, you can see a TradeStation chart that shows all the information we're going to use in our machine learning project:

NOTE

In my experience, I've noticed that traders and data scientists approach the financial market differently. Traders focus more on charts and less on numbers, while data scientists look at a lot of numbers and fewer charts. For traders, a chart is a decision-making tool, whereas for data scientists, it's just a (sometimes inconvenient) way to represent an idea. In this project, we will maintain a hybrid approach.

Figure 11.6 – A TradeStation chart showing a target and its features

If you're a trader, you will appreciate the chart shown in *Figure 11.6*, which shows the targets (the green bars), the features (two SMAs – Feature1), and the RSI (Feature2).

Conversely, if you're a data scientist, you might need to look at the dataset, writing the following code to print it out:

```
var:Target_(0);
If H>H[1] and L>L[1] then Target_=1 else Target_=0;
var:Feature1(0),Feature2(0);
if average(close,20)[1]>average(close,200)[1] then Feature1=1 else Feature1=0;
if RSI(Close,3)[1]>50                         then Feature2=1 else Feature2=0;
print(date,",",Feature1,",",Feature2,",",Target_);
```

Applying the preceding code to the AAPL (Apple Inc.) daily chart, we can see the dataset printed in the TradeStation Print Log (*Figure 11.7*):

TradeStation - Untitled Desktop: 1 - Chapter10 ML - [Chart Analysis - AAPL Daily [NASDAQ] Apple Inc]
File Edit View Window Help
Type a symbol/command
Apps Trade Position Graph
Timeframe Drawing Studies Style Data Settings
SA AAPL - Daily NASDAQ L=209.04 -4.92% Mov Avg 2 Lines (Close , Close ,20,200, 0) 223.79 189.68 Chapter10_ML_Targets () 225.60 217.71
RSI (Close ,5, 30 ,30,... 20.47 30.00 30.00
230.00
223.79
217.71
200.00
189.68
100.00
60.00
20.47
28 Jun'24 10 17 24 Jul'24 8 15 22 29 Aug'24 12
1240725.00, 1.00, 1.00, 0.00
1240726.00, 1.00, 1.00, 0.00
1240729.00, 1.00, 1.00, 0.00
1240730.00, 1.00, 0.00, 1.00
1240731.00, 1.00, 0.00, 1.00
1240801.00, 1.00, 0.00, 0.00
1240802.00, 1.00, 0.00, 1.00
Print Log
Chapter10 ML
IM1275491M Open Positions 6 Buying Power $3,927,174 Open P/L ($7,356) Closed P/L $0 05 Aug 2024 04:44 09 PM Sim Data

Figure 11.7 – The dataset in the TradeStation Print Log

As shown in *Figure 11.7*, the dataset is composed of four comma-separated columns – the date, Feature1, Feature2, and the target.

Training/testing data split

Every machine learning model needs a **training phase** and a testing phase. During the training phase, the model learns from the provided dataset by adjusting its parameters to minimize errors and improve its predictions. During the **testing phase**, the model uses the mathematical and statistical relationships it discovered during the training phase and is tested on unseen data.

This is why the available dataset is divided into two parts – the training dataset and the testing dataset. This division, within financial price datasets, generally occurs in chronological order. For example, in the case we will study, we will use the first 7 years of historical data of APPL to train a model; the last 3 years will be used for testing (see *Figure 11.8*):

Figure 11.8 – The dataset training/testing split

In *Figure 11.8*, in the green rectangle on the left, you can see the 7-year training historical period, starting from July 1, 2014, and ending on July 31, 2021; in the red rectangle on the right, you can see the 3-year testing period, starting from August 1, 2021, and ending on July 31, 2024.

Model selection

So far, we've identified the features of our model, which in this specific case are the trend, represented by two SMAs, and the pullback, represented by an RSI above a threshold. We've also identified the target, which is the H>H[1] and L>L[1] pattern. Now, it's time to identify the mathematical relationship that exists between the features and the target.

If we were Python programmers, we could now enjoy using the numerous library alternatives, but our goal is not to write a beautiful program. Our goal is to make money in the market while all other participants are trying to take away our capital. Therefore, we will settle for simpler mathematical concepts that are no less effective.

Let's start with the reasoning – we have two variables (the trend and the pullback), and we know that they both positively influence the price of the financial asset. Both features can only take binary values; similarly, the target, as we have seen, can only take binary values.

The first idea that comes to mind is to create a very simple function that averages the two variables according to the following formula:

$$S = \frac{W1 \text{ x Feature1} + W2 \text{ x Feature2}}{W1 + W2}$$

In this function, the *W1* and *W2* variables represent weights assigned to *Feature1* and *Feature2*, respectively, to calibrate their importance in recognizing the target. The function assumes values between 0 and 1; it will be 0 when neither feature is present on the previous day, and it will be 1

when both features were present on the previous day. It will assume values between 0 and 1 based on the weights that are assigned through simulations.

> ***NOTE***
>
> *The formula we used, S, is a way to replicate and simplify the sigmoid function. A sigmoid refers to an S-shaped curve, commonly used in mathematics and machine learning. Its shape rises steeply and then flattens out at the top and bottom, resembling an "S." The most common sigmoid function is the* ***logistic function****, which is defined as* $S(x) = 1/(1 + e^{x})$.
>
> *Here, σ(x) is the output, ranging from 0 to 1, and e is the base of the natural logarithm (approximately, this is 2.718).*

Model training

Now that we've defined a mathematical relationship between the features and the target, we can start training the model. For the training phase, we'll use the following code:

```
var:Target_(0);
If H>H[1] and L>L[1] then Target_=1 else Target_=0;
Input:SmaLen1(20),SmaLen2(200),RsiLen(5),RsiTrig(50);
var:Feature1(0),Feature2(0);
if average(close,SmaLen1)[1]>average(close,SmaLen2)[1] then Feature1=1 else Feature1=0;
if RSI(Close,RsiLen)[1] > RsiTrig                      then Feature2=1 else Feature2=0;
input:W1(1),W2(0);
var:S(0);
S = (W1 * Feature1 + W2 * Feature2 )/(W1+W2);
var:y_pred(0);
Input: STrig(0.5);
if S>Strig then y_pred=1 else y_pred=0;
//Confusion matrix
var:TP(0),FP(0),FN(0),TN(0),Tot(0),Accuracy(0);
if y_pred=1 and Target_=1 then TP=TP+1;
if y_pred=1 and Target_=0 then FP=FP+1;
if y_pred=0 and Target_=1 then FN=FN+1;
if y_pred=0 and Target_=0 then TN=TN+1;
tot=TP+FP+FN+TN;
Accuracy=(TP+TN)/Tot;
```

In the preceding code, the **`S = (W1 * Feature1 + W2 * Feature2 )/(W1+W2);`** formula calculates the **`S`** function, assigning the **`W1`** weight to **`Feature1`** and the **`W2`** weight to **`Feature2`**, obtaining a weighted average of the two features' readings.

The **`if S>Strig then y_pred=1 else y_pred=0;`** command provides the prediction (**`y_pred`**) for the following day; you can see that we assume the **`S`** function to be higher than the **`Strig`** level to predict a positive target.

Finally, **`Accuracy=(TP+TN)/Tot;`** calculates the accuracy of the model according to the confusion matrix.

> ***NOTE***
>
> *A **confusion matrix** is a table that's used to evaluate a classification model's performance by showing counts of true positives, true negatives, false positives, and false negatives. Key metrics such as accuracy, calculated as* $Accuracy = \frac{\text{True Positives + True Negatives}}{\text{Total Instances}}$*, are derived from this matrix. We will revisit this later in this chapter.*

Now, let's run the code with the settings shown in *Table 11.3* applied to an APPL (Apple Inc.) chart, with a history ranging from July 1, 2014, to July 31, 2024 (a 7-year training period):

Input	**Value**
SmaLen1	20
SmaLen2	200
RsiLen	5
RsiTrig	50
W1	1
W2	1
STrig	0.5

Table 11.3 – The first-run settings

By running the strategy with the settings from *Table 11.3*, the TradeStation Print Log will provide the results shown in *Table 11.4*:

SmaLen1	**SmaLen2**	**RsiLen**	**RsiTrig**	**Strig**	**TP**	**TN**	**Tot**	**Accuracy**
20	200	5	50	0.5	398	473	1563	56%

Table 11.4 – The first-run results

As shown in *Table 11.4*, by applying the S formula, the model can properly predict the presence of the target pattern for the following day in 398 instances. Simultaneously, it can correctly predict the absence of the target pattern in 473 instances. So, we can say that the accuracy of the model in recognizing the target pattern is (398+473)/1563 = 56%.

This is an encouraging result, but it could be due to chance. Let's try performing some sensitivity analysis to understand the features' importance (see *Figure 11.9*):

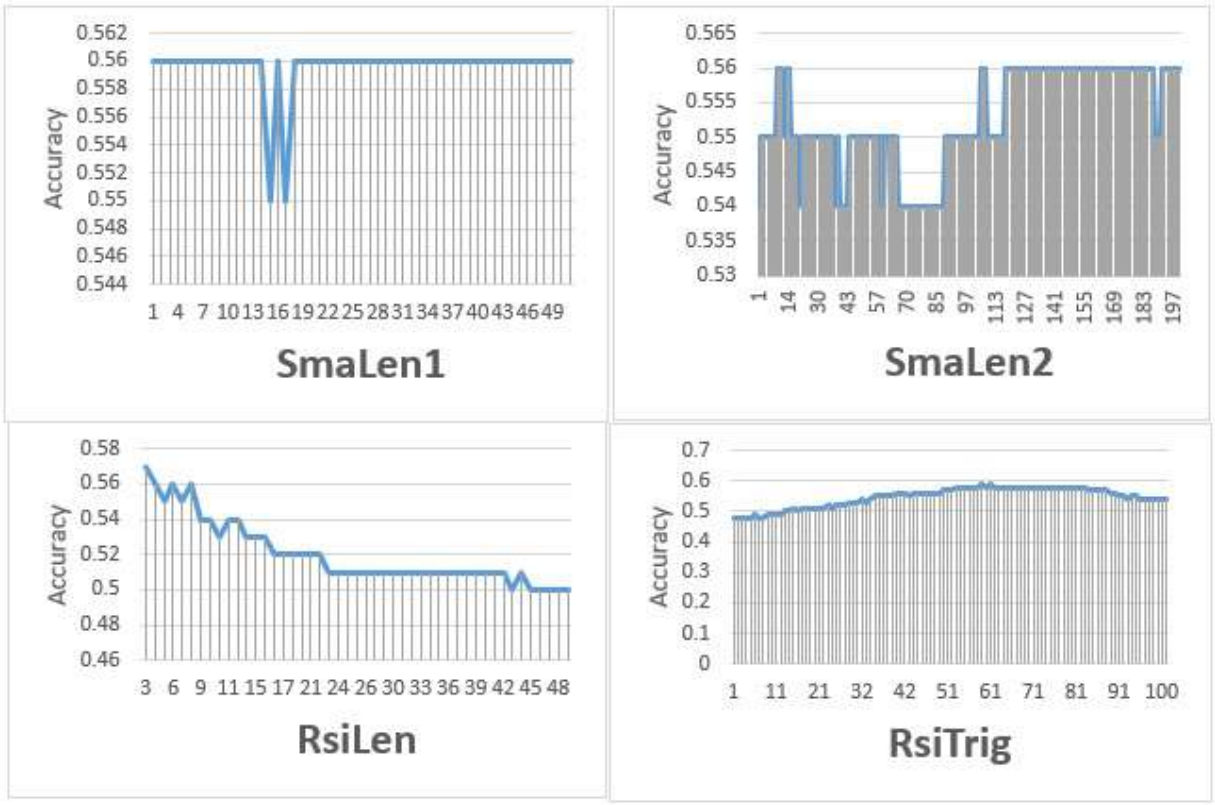


Figure 11.9 – The features' sensitivity analysis

In *Figure 11.9*, you can see four sensitivity analyses, all of which were obtained by running the TradeStation optimization process for the training period of AAPL (Apple Inc.). Note that even though we're using the same TradeStation functions we learned about in the previous chapters (namely, the backtesting engines), in this case, we're using the strategy code not to simulate the financial performance of the strategy but to evaluate its predictive capabilities. The output of each optimization is an **accuracy** value, which allows us to understand how well the model can recognize the identified target, which, as you'll recall, is given by the **H>H[1] and L>L[1]** pattern. All the results that have been obtained so far refer to the in-sample data of the training period. Looking at the preceding figure, we can see that the accuracy of the model is quite stable – around 56%. However, the model's accuracy doesn't change by changing SmaLen1, and it increases with the length of SmaLen2, decreases with the length of the RSI, and has a maximum value for RsiTrig – around 60.

So, let's set SmaLen1 to 20, SmaLen2 to 200, RsiLen to 3, and RsTrg to 60. Next, we'll change the feature weights to test their importance. To achieve this, we'll run a new optimization, varying both W1 and W2 from 1 to 5. The results can be seen in *Figure 11.10*:

Accuracy	W2				
W1	1	2	3	4	5
1	59%	60%	60%	60%	60%
2	48%	59%	60%	60%	60%
3	48%	48%	59%	60%	60%
4	48%	48%	48%	59%	60%
5	48%	48%	48%	48%	59%

Figure 11.10 – Sensitivity analysis for W1 and W2

In *Figure 11.10*, you can see how the model's accuracy increases with the increase of the W2 weight and decreases with the increase of the W1 weight. This indicates that, in recognizing the up session pattern, the RSI indicator is more important than the two moving averages.

Model testing and evaluation

Once the model has been trained on a portion of the available data, with the training period for the model on AAPL (Apple Inc.) being 7 years, we can test it on some out-of-sample data, which consists of 3 years of test data. *Table 11.5* shows the test results:

SmaLen1	**SmaLen2**	**RsiLen**	**RsiTrig**	**Strig**	**W1**	**W2**	**TP**	**TN**	**Tot**	**Accuracy**
20	200	3	60	0.5	1	2	206	297	807	62%

Table 11.5 – The test results

As shown in *Table 11.5*, by applying the S formula to the 3 years of unseen data (testing period), the model has been able to correctly predict the presence of the target pattern for the following day in 206 instances (**TP**). Additionally, it correctly predicted the absence of the target pattern in 297 instances (**TN**). Therefore, the accuracy of the model in recognizing the target pattern on out-of-sample data is (206+297)/807 = 62%, indicating that we've succeeded since the accuracy is stable.

Model deployment

Now that we have a model that's been trained on 7 years and tested on 3 years of out-of-sample data, with a stable accuracy above 60%, we can move on to the deployment phase.

We're doing this at this stage, even though it's not exactly part of the machine learning pipeline, to check if we are on the right track from a financial perspective.

So, let's perform traditional backtesting to determine whether, in addition to pattern recognition accuracy, the model can generate profit during the out-of-sample testing period. To do this, we can write the following code:

```
Input:SmaLen1(20),SmaLen2(200),RsiLen(5),RsiTrig(50);
var:Feature1(0),Feature2(0);
if average(close,SmaLen1)>average(close,SmaLen2) then Feature1=1 else Feature1=0;
if RSI(Close,RsiLen) > RsiTrig                   then Feature2=1 else Feature2=0;
input:W1(1),W2(0);
var:S(0);
S = (W1 * Feature1 + W2 * Feature2 )/(W1+W2);
Input: STrig(0.5);
var:Qty(0),StopLev(0),TgtLev(0);
Qty = 10000/close;
If S>STrig then begin
    Buy Qty shares this bar close;
    StopLev=L;
    TgtLev=H;
End;
if mp =1 then begin
    sell next bar StopLev stop;
    sell next bar  qty/2  shares TgtLev limit;
End;
if barssinceentry=1 then sell this bar close;
```

Note that in the preceding code, all references to the target have disappeared. Additionally, the **`[1]`** command that we used in the previous code snippet no longer makes sense. Additionally, we used the **`S>STrig`** condition to open a new long position at the bar close and the **`If barssinceentry=1`** condition to hold the position for one day.

Figure 11.11 shows the backtesting result:

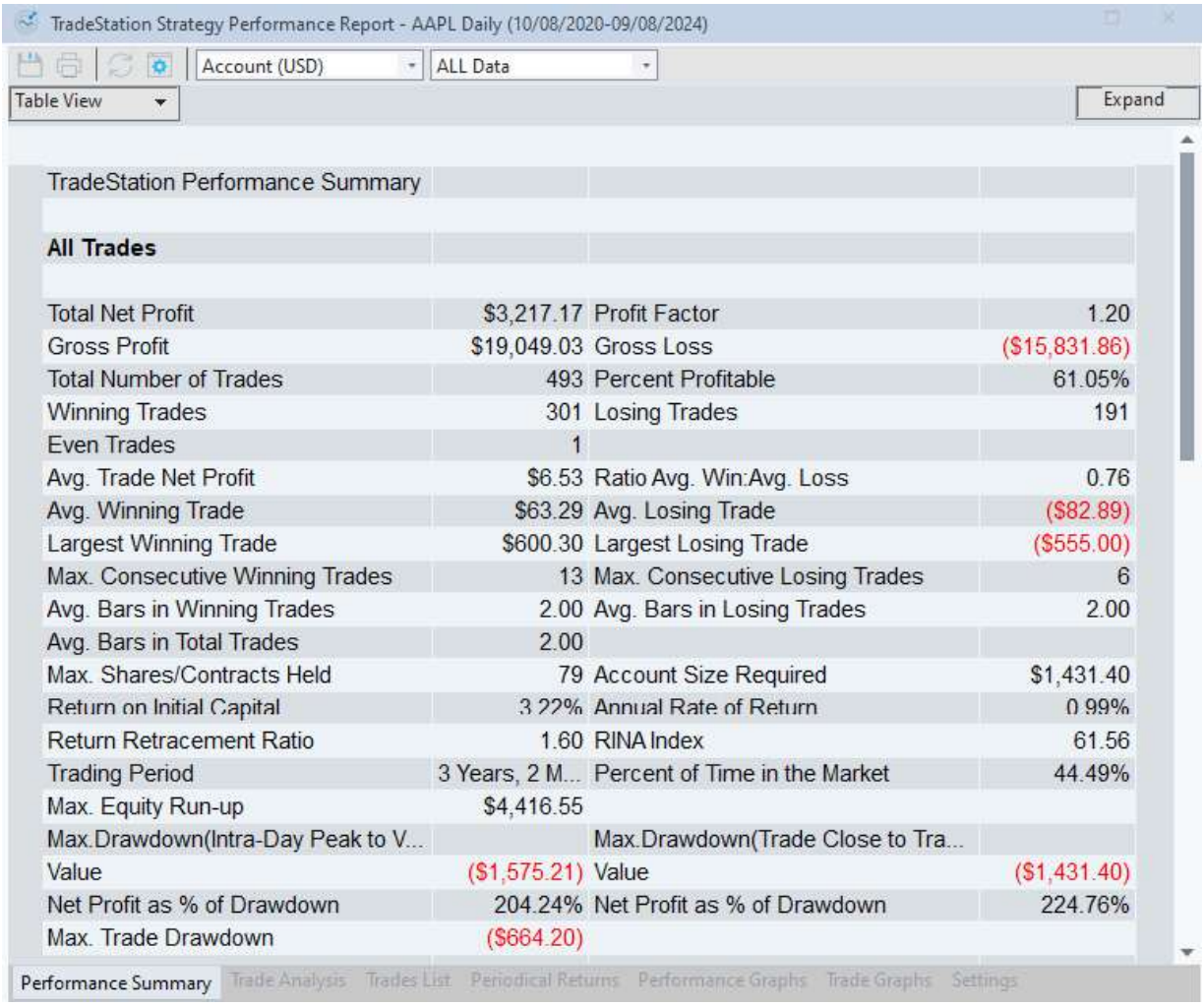

TradeStation Strategy Performance Report - AAPL Daily (10/08/2020-09/08/2024)
Account (USD)
ALL Data
Table View
Expand
TradeStation Performance Summary
All Trades
Total Net Profit $3,217.17 Profit Factor 1.20
Gross Profit $19,049.03 Gross Loss ($15,831.86)
Total Number of Trades 493 Percent Profitable 61.05%
Winning Trades 301 Losing Trades 191
Even Trades 1
Avg. Trade Net Profit $6.53 Ratio Avg. Win:Avg. Loss 0.76
Avg. Winning Trade $63.29 Avg. Losing Trade ($82.89)
Largest Winning Trade $600.30 Largest Losing Trade ($555.00)
Max. Consecutive Winning Trades 13 Max. Consecutive Losing Trades 6
Avg. Bars in Winning Trades 2.00 Avg. Bars in Losing Trades 2.00
Avg. Bars in Total Trades 2.00
Max. Shares/Contracts Held 79 Account Size Required $1,431.40
Return on Initial Capital 3.22% Annual Rate of Return 0.99%
Return Retracement Ratio 1.60 RINA Index 61.56
Trading Period 3 Years, 2 M... Percent of Time in the Market 44.49%
Max. Equity Run-up $4,416.55
Max.Drawdown(Intra-Day Peak to V... Max.Drawdown(Trade Close to Tra...
Value ($1,575.21) Value ($1,431.40)
Net Profit as % of Drawdown 204.24% Net Profit as % of Drawdown 224.76%
Max. Trade Drawdown ($664.20)
Performance Summary
Trade Analysis
Trades List
Periodical Returns
Performance Graphs
Trade Graphs
Settings

Figure 11.11 – The backtesting result of the machine learning model

As shown in *Figure 11.11*, our model is not only able to predict the up session but also able to make a profit of 3,217,7 USD, with a trading amount of 10,000 USD in 3 years of out-of-sample data.

To deploy the model, we can just run the strategy we have built so far for our simulated account (or for our real account after a paper trading period) in real time.

The confusion matrix

I've chosen to dedicate an entire section of this chapter to the confusion matrix (don't be discouraged by the name) because I believe, among the various methods for evaluating model effectiveness, that the matrix closely aligns with a trader's way of thinking better than any other. It's straightforward, avoids complex statistical formulas, and is highly effective.

A **confusion matrix** is a table that's used to evaluate the performance of a classification model. It summarizes the model's predictions compared to actual outcomes by displaying four key components

(*Table 11.6*):

Component	**Description**
True positives (**TPs**)	Represents correct positive predictions
True negatives (**TNs**)	Represents correct negative predictions
False positives (**FPs**)	Represents incorrect positive predictions
False negatives (**FNs**)	Represents incorrect negative predictions

Table 11.6 – Confusion matrix components

The matrix helps calculate various performance metrics. In *Table 11.7*, I have identified those that I consider most relevant:

Performance metric	**Description**	**Formula**
Accuracy	Measures the proportion of correctly classified instances (both positive and negative)	$\frac{TP + TN}{TP + TN + FP + FN}$
Precision	Indicates how many of the positive predictions were correct	$\frac{TP}{TP + FP}$
Recall	Measures how many of the actual positives were correctly identified	$\frac{TP}{TP + FN}$
Specificity	Measures how many of the actual negatives were correctly identified	$\frac{TN}{TN + FP}$

Table 11.7 – Performance metrics

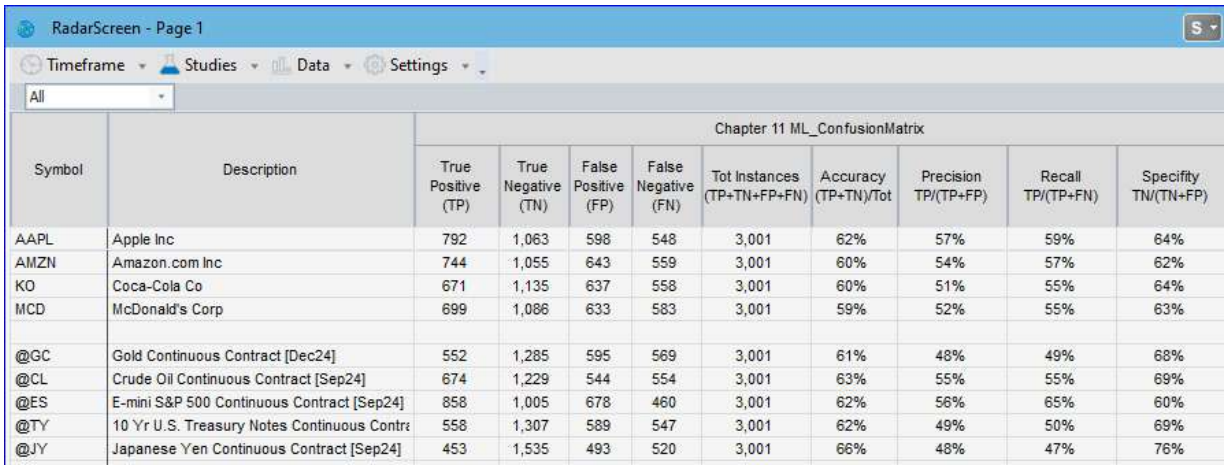
Let’s write the performance metrics shown in *Table 11.7* in EasyLanguage:

```
var:Target_(0);
If H>H[1] and L>L[1] then Target_=1 else Target_=0;
Input:SmaLen1(20),SmaLen2(200),RsiLen(5),RsiTrig(50);
var:Feature1(0),Feature2(0);
if average(close,SmaLen1)[1]>average(close,SmaLen2)[1] then Feature1=1 else Feature1=0;
if RSI(Close,RsiLen)[1] > RsiTrig                      then Feature2=1 else Feature2=0;
input:W1(1),W2(0);
var:S(0);
S = (W1 * Feature1 + W2 * Feature2 )/(W1+W2);
var:y_pred(0);
Input: STrig(0.5);
```

```
if S>Strig then y_pred=1 else y_pred=0;
var:TP(0),FP(0),FN(0),TN(0),Tot(0);
if y_pred=1 and Target_=1 then TP=TP+1;
if y_pred=1 and Target_=0 then FP=FP+1;
if y_pred=0 and Target_=1 then FN=FN+1;
if y_pred=0 and Target_=0 then TN=TN+1;
tot=TP+FP+FN+TN;
var:Accuracy(0),Precision(0),Recall(0), (0;
if tot>0   then Accuracy=(TP+TN)/Tot;
if TP+FP>0 then Precision = TP/(TP+FP);
if TP+FN>0 then Recall=TP/(TP+FN);
if TN+FP>0 then Specifity=TN/(TN+FP);
plot1(TP,"True Positive (TP)");
plot2(TN,"True Negative (TN)");
plot3(FP,"False Positive (FP)");
plot4(FN,"False Negative (FN)");
plot5(Tot,"Tot Instances (TP+TN+FP+FN)");
plot11(Accuracy,"Accuracy (TP+TN)/Tot");
plot12(Precision,"Precision TP/(TP+FP)");
plot13(Recall,"Recall TP/(TP+FN)");
plot14(Specifity,"Specifity TN/(TN+FP)");
```

In the preceding code, we just programmed the confusion matrix formulas of *Table 11.7* in EasyLanguage.

Figure 11.12 shows the preceding code applied to a TradeStation RadarScreen:



Symbol	Description	Chapter 11 ML_ConfusionMatrix								
		True Positive (TP)	True Negative (TN)	False Positive (FP)	False Negative (FN)	Tot Instances (TP+TN+FP+FN)	Accuracy (TP+TN)/Tot	Precision TP/(TP+FP)	Recall TP/(TP+FN)	Specifity TN/(TN+FP)
AAPL	Apple Inc	792	1,063	598	548	3,001	62%	57%	59%	64%
AMZN	Amazon.com Inc	744	1,055	643	559	3,001	60%	54%	57%	62%
KO	Coca-Cola Co	671	1,135	637	558	3,001	60%	51%	55%	64%
MCD	McDonald's Corp	699	1,086	633	583	3,001	59%	52%	55%	63%
@GC	Gold Continuous Contract [Dec24]	552	1,285	595	569	3,001	61%	48%	49%	68%
@CL	Crude Oil Continuous Contract [Sep24]	674	1,229	544	554	3,001	63%	55%	55%	69%
@ES	E-mini S&P 500 Continuous Contract [Sep24]	858	1,005	678	460	3,001	62%	56%	65%	60%
@TY	10 Yr U.S. Treasury Notes Continuous Contr	558	1,307	589	547	3,001	62%	49%	50%	69%
@JY	Japanese Yen Continuous Contract [Sep24]	453	1,535	493	520	3,001	66%	48%	47%	76%

Figure 11.12 – The confusion matrix in a TradeStation RadarScreen

In *Figure 11.12*, we applied the preceding code to a TradeStation RadarScreen that listed four stocks (AAPl, AMZN, KO, and MCD) and five futures (gold, crude oil, S&P500, Treasury note, and yen). By doing so, we calculated the confusion matrix with the same purpose – recognizing an up session for the coming day. Let's analyze the RadarScreen output from both a machine learning and market perspective:

- **Accuracy**: This measures the proportion of correctly classified instances, and it shows the lowest reading (60%) for AMZN (Amazon.com Inc.) and KO (Coca-Cola Co) and the highest reading for the Japanese Yen future (66%)
- **Precision**: This indicates how many of the positive predictions were correct, and it shows the lowest reading for the yen and gold futures (48%) and the highest reading (57%) for AAPL (Apple Inc.)

- **Recall**: This measures how many of the actual positives were correctly identified, and it shows the lowest reading (47%) for the Yen future and the highest reading for the S&P 500 (65%)
- **Sensitivity**: This measures how many of the actual negatives were correctly identified, and it shows the lowest reading for the S&P 500 (60%) and the highest reading for the Japanese Yen future (76%)

From a machine learning perspective, we can say that the model is relatively stable as its accuracy consistently hovers around 60%. If we analyze these results more closely, we can see that, while the accuracy remains constant across different instruments, this consistency is due to varying factors. For example, in some instruments, such as the S&P 500, the high accuracy is attributed to high recall, whereas in other futures contracts, it is due to high sensitivity. What this means is that the model has been successful in identifying the presence of the pattern in the following day's stock markets, while in other futures contracts, the model has been effective in recognizing the absence of the pattern. If we cross-reference this information with the pattern frequency shown in *Table 11.11*, we'll see that the high recall observed in stock markets could partly be due to long bias, while the high sensitivity in Japanese yen futures might be attributable to short bias in the instrument. In other words, even when using machine learning tools, as we've seen in algorithmic trading, we must play the role of the devil's advocate, always ensuring that the predictive results are not influenced by the structure of the data; otherwise, our efforts will be in vain. For this reason, let's move on to Project N.2.

Project N.2—recognizing down sessions

After learning how to use a machine learning model to identify an up session, it is essential to analyze the other side of the coin, which is the more hidden and potentially riskier aspect – the short side. Our goal now will be to recognize a down session (see *Figure 11.13*).

In this section, we're going to do the following:

- Select the target
- Select the features
- Perform the machine learning pipeline

Selecting the target

The objective of Project N.2 is to automatically recognize the **down session** (*Figure 11.13*) using the machine learning process we've explored so far:

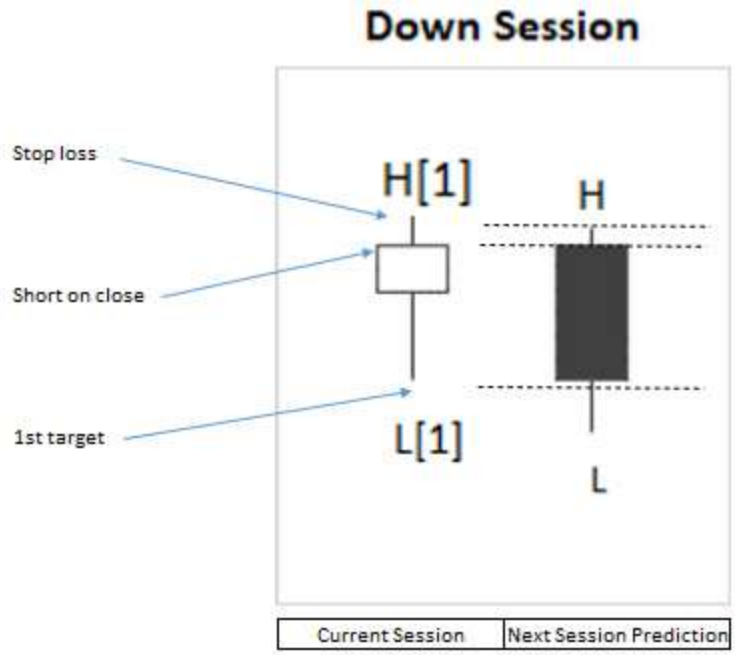


Figure 11.13 – A down session

In *Figure 11.13*, you can see the target pattern down session for Project N.2. Forecasting a down session allows the trader to plan a risk-calculated short trade such as the following:

- Sell short at the bar close
- Place a stop loss above today's high
- Place a first target on today's low
- Hold the position for one day

Let's look at a practical example. We'll assume that our model suggests the positive down session target for the following day. If the distance from today's high to the close is 2% and the distance from the close to the low is 1%, we can easily calculate the expected reward/risk ratio of our short trade. The reward-to-risk ratio will be as follows:

$$RewRisk\ ratio = \frac{1\%}{2\%} = 0.5$$

So, the reward-to-risk ratio is 0.5, which is quite low, so we can consider not investing in the market.

Selecting features

Let's stick to our previous model. So, for this second project, once again, we're going to feature two SMAs and an RSI. *Table 11.8* shows what our dataset will look like:

ID (Date)	**Feature1**	**Feature2**	**Target**

	FastSma < SlowSma	RSI < Threshold	H<H[1] and L < L[1]
1	0	0	1
2	1	1	1
3	1	0	0

Table 11.8 – The dataset

Table 11.8 shows the structure of our dataset, which consists of two features and one target:

- **Feature1**: This is a binary field represented by a FastSma below a SlowSma; it will be 1 in this case, as the FatsSma is above the SlowSma, and 0 otherwise
- **Feature2**: This is a binary field represented by RSI below a threshold; it will be 1 if the RSI is above the threshold and 0 if it is below it
- **Target**: This is a binary field as well, represented by the price conditions H<H[1] and L<L[1]; it will be 1 if today's high is higher than yesterday's high and today's low is above yesterday's low

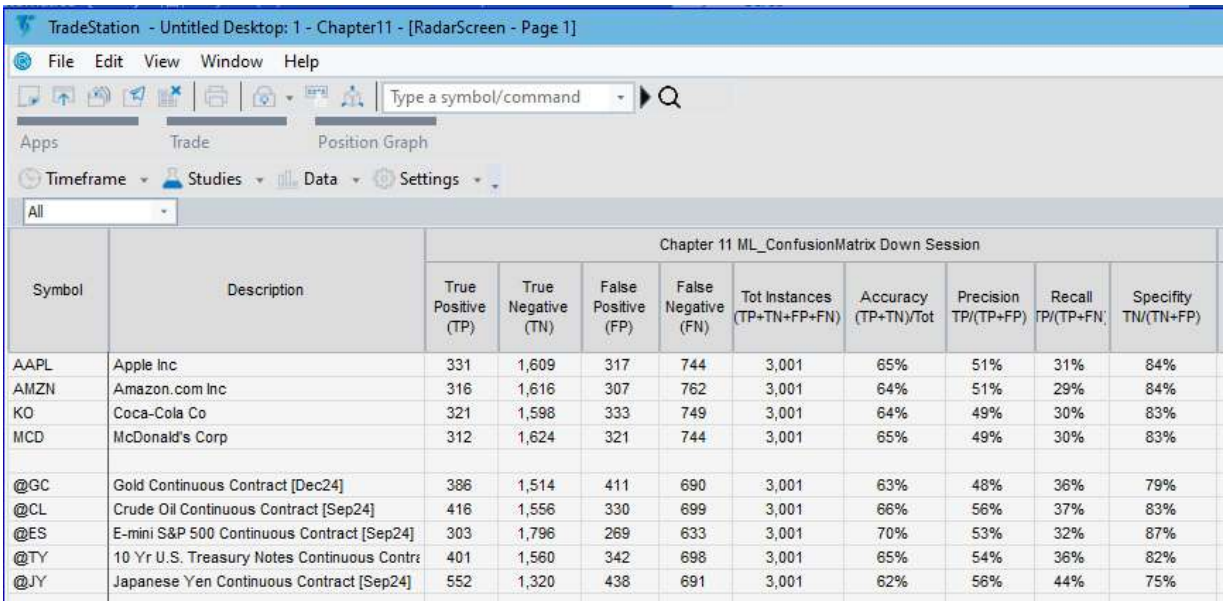
Final results

We'll assume all the steps in the pipeline, shown in *Figure 11.5*, are correct and proceed to present the final results of the analysis.

All the previous EasyLanguage code needs to be modified for the target and feature calculations, as follows:

```
var:Target_(0);
If H<H[1] and L<L[1] then Target_=1 else Target_=0;
Input:SmaLen1(20),SmaLen2(200),RsiLen(5),RsiTrig(50);
var:Feature1(0),Feature2(0);
if average(close,SmaLen1)[1]<average(close,SmaLen2)[1] then Feature1=1 else Feature1=0;
if RSI(Close,RsiLen)[1]<RsiTrig                        then Feature2=1 else Feature2=0;
```

As you can see, the formula to identify the target will now be `H<H[1] and L<L[1]`, while we need to adjust the features with `<` as we need both SMAs and RSI on the short side. *Figure 11.14* shows the confusion matrix:



Symbol	Description	Chapter 11 ML_ConfusionMatrix Down Session								
		True Positive (TP)	True Negative (TN)	False Positive (FP)	False Negative (FN)	Tot Instances (TP+TN+FP+FN)	Accuracy (TP+TN)/Tot	Precision TP/(TP+FP)	Recall TP/(TP+FN)	Specifity TN/(TN+FP)
AAPL	Apple Inc	331	1,609	317	744	3,001	65%	51%	31%	84%
AMZN	Amazon.com Inc	316	1,616	307	762	3,001	64%	51%	29%	84%
KO	Coca-Cola Co	321	1,598	333	749	3,001	64%	49%	30%	83%
MCD	McDonald's Corp	312	1,624	321	744	3,001	65%	49%	30%	83%
@GC	Gold Continuous Contract [Dec24]	386	1,514	411	690	3,001	63%	48%	36%	79%
@CL	Crude Oil Continuous Contract [Sep24]	416	1,556	330	699	3,001	66%	56%	37%	83%
@ES	E-mini S&P 500 Continuous Contract [Sep24]	303	1,796	269	633	3,001	70%	53%	32%	87%
@TY	10 Yr U.S. Treasury Notes Continuous Contra	401	1,560	342	698	3,001	65%	54%	36%	82%
@JY	Japanese Yen Continuous Contract [Sep24]	552	1,320	438	691	3,001	62%	56%	44%	75%

Figure 11.14 – The confusion matrix for the down session model

Let's analyze the results of *Figure 11.14*, where **accuracy** appears to be quite stable and above 62% (the lowest value of Japanese yen). By performing a comprehensive analysis of this data, we can see that the primary factor influencing accuracy is the **specificity** values, which are above 75% for all listed instruments. This means that the model is better at predicting the absence of a pattern than its presence; this would make sense for stock markets because of long bias (in fact, the highest sensitivity value is recorded for the S&P 500). However, there remains some uncertainty regarding the other markets.

From this, we can conclude that, although the model appears stable, there are aspects it doesn't explain, such as the high specificity across all markets. This indicates that even for this model, our work is only just beginning. By following this methodology, you can continue your own research and explore these issues further.

Summary

This chapter explored the practical application of AI in financial trading, focusing on machine learning for pattern recognition. It highlighted effective techniques and using EasyLanguage within TradeStation to integrate advanced learning algorithms into trading strategies. Despite EasyLanguage's limitations compared to Python, it remains a valuable tool for traders.

In this chapter, we learned that, unlike algorithmic trading, the primary goal of machine learning for pattern recognition is not to evaluate strategies from a financial perspective but, rather, to assess their predictive capabilities. Evaluating the financial performance of a strategy without verifying its predictive abilities first is a waste of time and often leads to overfitting.

This chapter drew inspiration from Sir R. Fisher's pioneering pattern recognition experiments of the 1930s, which share many elements with trading. The key topics that were covered included defining machine learning for pattern recognition, exploring Fisher's Iris dataset, classifying trading sessions, and looking at two hands-on projects focused on identifying up session and down session patterns. The aim is to equip you not only with practical tools and techniques but also with the mindset needed to independently explore and extend this fascinating area of research, using accessible programming methods.

The future paths you can take by following the work model suggested in this chapter are countless – you can change the targets, customizing them to your experience and trading style, or you can choose from the numerous feature alternatives that technical analysis provides. The primary goal is to move away from the logic of algorithmic backtesting, which is based solely on financial metrics, and focus primarily on the robustness of the models instead.

In this book, we delved into EasyLanguage programming and the comprehensive use of the TradeStation platform, covering everything from basic concepts to advanced AI-related topics. We explored managing algorithmic trading across various financial markets and controlling trade risk. We examined how to implement what we've learned in live trading scenarios and developed a scientific mindset toward financial markets. We discovered that financial metrics can be of limited value if the strategies you develop, beyond backtesting, lack predictive power. Additionally, we found that machine learning can be a powerful tool to assess and enhance the predictive capability of your strategies, even if you're not a data scientist and don't have a master's degree in Python.

The main advice I can offer to those who have had the patience to reach the final pages of this book is as follows. The financial sector is full of people who want to teach you things they don't know or have never experienced. Be smart, and believe only in what you can see and touch with your own hands, but most importantly, believe in yourself.

In your journey through the "financial markets sea," you may encounter "pirates" –people who aim, in a sophisticated way, to steal money from others. Stay away from them; being a pirate isn't a job for everyone.

Work as part of a team; the psychological aspect of this job is crucial, and overcoming motivational drawdowns is much easier when you're surrounded by supportive teammates – just make sure there are no "pirates" among them.

It's a world where theory and practice are light-years apart, you have to get your hands dirty, put yourself on the line, and risk a portion – ideally, the smallest possible – of your capital.

Simplicity is key, even in complex topics such as those discussed in the preceding chapters, and above all, remember that not everything that glitters is gold.

Index

As this ebook edition doesn't have fixed pagination, the page numbers below are hyperlinked for reference only, based on the printed edition of this book.

A

accuracy value 240

AI predictions

  embedding, into EasyLanguage indicators 212

  embedding, into EasyLanguage strategies 213

algorithmic trading 1, 2

  definition 2, 3

  developing 52, 53

  in quantitative hedge funds 3

AND logical operator

  using 41, 42

Apple Inc. (AAPL) 144

artificial intelligence (AI) 207, 225

Average Directional Index (ADX) 126, 127

average true range (ATR) 148

average true range percent (ATRpct) 148

B

backtesting 77, 78

  re-running 96

  versus buy-and-hold strategy 97, 98

base currency 173

breakout 107, 179

  algorithms, writing 181-183

British currency (GBP) 183

buy-and-hold strategy 97

versus backtesting 97, 98

C

chart analysis 8-13

confusion matrix 239, 243-246

consumer price index (CPI) 174

continuous futures contract 164

D

data collection 233

feature formula 234-236

target formula 234

double-way sensitivity analysis 80, 85

Excel Optimization Report 86-90

TradeStation Optimization Report 85

E

EasyLanguage 18, 19, 24, 25

bridging 210-212

code 77

expressions 27, 28

fundamentals, exploring 23

indicators, writing 29, 30

indicators, used for embedding AI predictions 212

punctuation 28, 29

statements 28

usage 25, 26

versus Python 208, 209

words 27
EasyLanguage, indicators
AND logical operator, using 41, 42
Close 31
Close Open 32
Crossover 38, 39
MidPrice 33, 34
Momentum 36, 37
Moving Average 37, 38
NetCh 34
NetCh, with variables 35
Real Body 32, 33
reference time 40
Uptrend 39, 40
EasyLanguage order syntax
components 43
EasyLanguage strategies
used, for embedding AI predictions 213
Volatility Predictor model as filter, using 213
Volatility_Predictor model as filter, using 214-216
Volatility Predictor model for money management, using 217, 218
Volatility Predictor model for position sizing, using 218, 219
equal dollar risk sizing 151
equal dollar risk technique
used, for incorporating technical exits 156
without technical exit levels 153, 154
with technical exits 157
writing 154-156

writing, in technical exits 157, 158
EquityEdge Buy&Hold Analyzer 97
equity lines chart on Excel 94
backtesting, re-running 95, 96
print instruction, modifying 94
ESU24 symbol 192
European currency (EUR) 183
Excel Optimization Report 86-90
Exchange-Traded Fund (ETF) 59
existing trend 124
Average Directional Index (ADX) 126, 127
Supertrend indicator 124-126
exit tools
stop loss 144
target 144
trailing stop 144
expiration date 163
Extreme Gradient Boosting (XGBoost) 209

F

fake breakout 180
algorithms, writing 183-187
false breakout
versus breakout 179
final impulse 131
close above/below 132
oversold/overbought 132
financial information 54, 55
internal budgeting 55

reporting cycle 55
financial rhythm 55
Fisher’s project 226
foreign exchange (Forex) 51, 173
foreign exchange (Forex), concepts 173
currency pairs 173
economic indicators 174
interest rate 173
major crosses, versus minor crosses 177, 178
margin trading 175
trading sessions 176
Forex algorithmic trading 173
algorithms, writing 178
forex market algorithms
intraday forex volatility, analyzing 178, 179
trading session overlaps, analyzing 178, 179
writing 178
forex rollover 175
front contract 163
front-month contract 163
fully automated trading plan, for futures 191
E-mini S&P 500 191
four futures portfolio, sizing 192-196
fully automated strategies, managing 198, 199
futures symbols, managing 191
gold 191
minimum capital required 196
risk management 197

T-note 191

yen 191

fundamental information 54

futures continuous contract 165

futures contracts 162

futures contracts, algorithmic trading 162, 163

time breakout strategies 168

futures contracts, concepts 163

continuous futures contract 164-166

expiration date 163, 164

futures sessions 167

margin trading 166

rollovers 163

futures rollover 175

futures sessions 167

G

Gandalf Project

reference link 209

gap 155

Gross domestic product (GDP) 174

H

higher highs 66

50-day high higher than 10 days before 69, 70

200-day high higher than 20 days before 66-68

I

in-sample data 99

EasyLanguage scripts, modifying 99

validation example 100, 101

Intel Corp. (INTC) 146

Iris dataset 226

L

logistic function 238

London session 176

long bias 230

long reversal strategy

backtesting 109

long reversal strategy, backtesting

data, exporting into Excel 111, 112

full Dow Jones 30 index 114, 115

multiple sensitivity analysis, running on HD 111

parameter set, selecting 113

ROA heatmap, creating 111, 112

stock starting point, establishing 110

M

machine learning

for pattern recognition 226

machine learning pipeline

building 233

data collection 233

model deployment 242, 243

model evaluation 241

model selection 237

model testing 241, 242

model training 238-241

testing phase 236, 237
training phase 236, 237
machine learning project, down session 247
features, selecting 248
final results 249, 250
target, selecting 247, 248
machine learning project, up session 231
features, selecting 232
machine learning pipeline, building 233
target, selecting 231, 232
maintenance margin level 167
major crosses
versus minor crosses 177, 178
major pairs 177
margin 175
margin call 167
margin rates
reference link 162
margin trading 166, 175
market noise 70
entry confirmation 71, 72
handling 70
volatility band 72-75
market participants
institutional investors 56
retail investors 56
market rationale 53
financial information 54, 55

prices and volumes 56
MaxBarsBack 45, 46
minor crosses
versus major crosses 177, 178
minor pairs 177
model deployment 242, 243
model evaluation 241
model selection 237
model testing 241
model training 238-241
Momentum function 36, 37
money management 143, 144
percent-based exit tools 146-148
price-based exit tools 144-146
volatility-based exit tools 148-151
moving averages 58
Close above SMA 58-61
fast SMA above slow SMA 62, 63
Rising SMA 64, 65
MultiCharts
URL 5
multiple-output function
input/output parameter 130
input parameter 130
multiple-way sensitivity analysis 80, 90

N

Negative Directional Indicator (-DI) 126
New York session 176

New York Stock Exchange (NYSE) 92

non-farm payrolls (NFPs) 174

O

one-way sensitivity analysis 80-85

operational plan

  loose plan 190

  tight plan 190

operational rhythm 55

optimal level of complexity 79

out-of-sample analysis 141, 142

out-of-sample data 99

  EasyLanguage scripts, modifying 99

  validation example 100, 101

overfitting

  strategy complexity 78

  strategy robustness 78

overnight interest 175

oversold 129

over the counter (OTC) 173

P

pattern recognition 226

percent-based exit tools 146-148

pipeline 233

P&L equation 152

position sizing 143, 151, 192

  equal dollar risk technique, without technical exit levels 153

  P&L equation 152

technical exits, incorporating in equal dollar risk technique 156
Positive Directional Indicator (+DI) 126
prevalence 230
price-based exit tools 144-146
print instruction
modifying 94
Procter & Gamble Co (PG) 149
profit and loss (P&L) 152
program analysis tools 3
program mechanical trading tools 3
program monitoring tools 3
program trading systems 3
pullback 128
Relative Strength Index (RSI) 128, 129
Stochastic oscillator 130, 131
pullback down 123
pullback up 122
Python
bridging 210-212
versus EasyLanguage 208, 209

Q

quantitative hedge funds 3
algorithmic trading 4
quote currency 173

R

RadarScreen 13-17
entries, tracking with 200-202

reforecast 55

Relative Strength Index (RSI) 128, 129, 232

Return on Account (ROA) 83, 109, 138, 183, 195

reversal 103, 121

  types 104

reversal down 105

reversal strategy

  existing trend 106

  final trend 107-109

  volatility compression 106, 107

  writing 106

reversal up 104, 105

risk management 143, 192

ROA report in Excel 92-94

rollover 166

RSI multiple-sensitivity analysis

  running, on HD 136-139

S

Scanner app 17, 18

semi-automated trading plan, on large stock lists 200

  entries, tracking with RadarScreen 200-202

  real positions, monitoring 204-206

  volatility-based sizes, calculating 203, 204

sensitivity analysis 79

  double-way sensitivity analysis 85

  multiple-way sensitivity analysis 90

  one-way sensitivity analysis 80-85

  RSI multiple-sensitivity analysis, running on HD 136-139

SMA multiple-sensitivity analysis, running on HD 140, 141
short strategies
backtesting 116
short strategies, backtesting
best parameter set, selecting 118-120
data, exporting into Excel 116-118
multiple-sensitivity analysis, running on SPY 116
ROA heatmap, creating 116-118
Simple Moving Average (SMA) 37, 58, 145, 201, 232
SMA multiple-sensitivity analysis
running, on HD 140, 141
SPDR S&P 500 ETF Trust (SPY) 116
Standard & Poor (S&P) 59
Stochastic oscillator 130, 131
raw line 130
smoothed line 130
stop loss 47-49
strategy complexity 78, 79
strategy robustness 78, 79
Supertrend indicator 124, 125
supervised learning 227
approach 227
swap 175
Sydney session 176
symbols backtesting 90
equity lines chart on Excel 94
ROA report in Excel 92-94
TradeStation stock symbol universe 90-92

T

take profit 46

target 232

technical information 54

technical-level exits 152

testing phase 236, 237

time breakout 168

time breakout, strategies

  script writing 169, 170

  sensitivity analysis, using 170-172

  time support and resistance, defining 168, 169

Tokyo session 176

TradeStation 4, 7

  chart analysis 8-13

  Desktops 21

  downloading 5

  EasyLanguage 18, 19

  history 4, 5

  installation procedure 6

  Optimization Report 85, 86

  overview 4

  platform 1

  RadarScreen 13-17

  Scanner app 17, 18

  stock symbol universe 90-92

  technical minimum requirements 6

  URL 5

  using, to collect predictions from multiple AI models 221, 222

workspace 20, 21

workspace 20

TradeStation strategy

EasyLanguage order syntax 43-45

MaxBarsBack 45, 46

programming 42, 43

stop loss levels 47-49

take profit 46

writing 42

trading operational plan resources

algorithms 190

brokers 190

market data 190

team 190

trading platform 190

trading session

labeling 228-230

trading sessions

London session 176

New York session 176

Sydney session 176

Tokyo session 176

training phase 236, 237

trend algorithmically

higher highs 66

identifying 56, 57

moving averages 58

trend pullback components 133-136

existing trend 124
final impulse 131
out-of-sample analysis 141, 142
pullback 128
sensitivity analysis 136
writing 124
trend pullback strategy
market rationale 122, 123
trend reversal 103
trends 121

U

unsupervised learning 227

V

volatility 148
volatility-based exit tools 148-151
volatility dashboard
creating, for multiple assets 220, 221
Volatility_Predictions function 213
Volatility Predictor 209

packtpub.com

Subscribe to our online digital library for full access to over 7,000 books and videos, as well as industry leading tools to help you plan your personal development and advance your career. For more information, please visit our website.

Why subscribe?

- Spend less time learning and more time coding with practical eBooks and Videos from over 4,000 industry professionals
- Improve your learning with Skill Plans built especially for you
- Get a free eBook or video every month
- Fully searchable for easy access to vital information
- Copy and paste, print, and bookmark content

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at packtpub.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at customercare@packtpub.com for more details.

At www.packtpub.com, you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on Packt books and eBooks.

Packt is searching for authors like you

If you're interested in becoming an author for Packt, please visit authors.packtpub.com and apply today. We have worked with thousands of developers and tech professionals, just like you, to help them share their insight with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Share Your Thoughts

Now you've finished *TradeStation EasyLanguage for Algorithmic Trading*, we'd love to hear your thoughts! If you purchased the book from Amazon, please click here to go straight to the Amazon

review page for this book and share your feedback or leave a review on the site that you purchased it from.

Your review is important to us and the tech community and will help us make sure we're delivering excellent quality content.

Download a free PDF copy of this book

Thanks for purchasing this book!

Do you like to read on the go but are unable to carry your print books everywhere?

Is your eBook purchase not compatible with the device of your choice?

Don't worry, now with every Packt book you get a DRM-free PDF version of that book at no cost.

Read anywhere, any place, on any device. Search, copy, and paste code from your favorite technical books directly into your application.

The perks don't stop there, you can get exclusive access to discounts, newsletters, and great free content in your inbox daily

Follow these simple steps to get the benefits:

1. Scan the QR code or visit the link below

https://packt.link/free-ebook/9781835881200

2. Submit your proof of purchase
3. That's it! We'll send your free PDF and other benefits to your email directly

Contents

1. TradeStation EasyLanguage for Algorithmic Trading
2. Contributors
3. About the author
4. About the reviewer
5. Preface
    1. Who this book is for
    2. What this book covers
    3. To get the most out of this book
    4. Download the example code files
    5. Conventions used
    6. Get in touch
    7. Share Your Thoughts
    8. Download a free PDF copy of this book
6. Chapter 1: Introduction to Algorithmic Trading and the TradeStation Platform
    1. Introducing algorithmic trading
        1. Algorithmic trading definition
        2. Algorithmic trading in quantitative hedge funds
    2. Introducing the TradeStation platform
        1. The TradeStation story
        2. Download and installation procedure
        3. Introducing TradeStation apps
        4. Workspaces and desktops
    3. Summary
7. Chapter 2: Getting Hands-On with EasyLanguage
    1. Understanding the basics of EasyLanguage
        1. What is EasyLanguage?
        2. How EasyLanguage works
        3. EasyLanguage words
        4. EasyLanguage expressions
        5. EasyLanguage statements
        6. EasyLanguage punctuation
    2. Writing indicators
        1. Exercise 01—Close

    2. Exercise 02—Close Open
    3. Exercise 03—Real Body
    4. Exercise 04—MidPrice
    5. Exercise 05—NetCh
    6. Exercise 06—NetCh with variables
    7. Exercise 07—Momentum
    8. Exercise 08—Moving Average
    9. Exercise 09—Crossover
    10. Exercise 10—Uptrend
    11. Exercise 11—Time
3. Writing strategies
    1. How to program a basic strategy
    2. EasyLanguage order syntax
    3. MaxBarsBack
    4. How to program take profit and stop loss levels
4. Summary

8. Chapter 3: Writing a Trend Strategy
    1. Developing trading ideas
    2. Market rationale behind a trend strategy
        1. Information
        2. Market participants
        3. Prices and volumes
    3. Identifying a trend algorithmically
        1. Moving averages
        2. Higher highs
    4. Handling market noise
        1. Entry confirmations
        2. Volatility bands
    5. Summary
9. Chapter 4: Strategy Backtesting and Validation
    1. Understanding backtesting and overfitting
        1. Strategy complexity
        2. Strategy robustness
    2. Sensitivity analysis
        1. One-way sensitivity analysis
        2. Double-way sensitivity analysis
        3. Multiple-way sensitivity analysis

    3. Backtesting across symbols
        1. TradeStation stock symbol universe
        2. ROA report in Excel
        3. Equity lines chart on Excel
    4. Backtesting versus buy-and-hold
    5. In-sample, out-of-sample analysis
        1. Modifying EasyLanguage scripts for in-sample, out-of-sample purposes
        2. An example of in-sample, out-of-sample validation
    6. Summary
10. Chapter 5: Reversal Strategies
    1. The market rationale behind a reversal strategy
        1. Reversal up
        2. Reversal down
    2. Writing a reversal strategy
        1. Existing trend
        2. Volatility compression
        3. Final trend
    3. Backtesting long strategies
        1. Identifying stock to start with
        2. Running a multiple sensitivity analysis on HD (Home Depot Inc.)
        3. Exporting data into Excel and creating an ROA heatmap
        4. Selecting the best parameter set
        5. Backtesting the strategy on the full Dow Jones 30 index
    4. Backtesting short strategies
        1. Running multiple-sensitivity analysis on the SPY
        2. Exporting data into Excel and creating an ROA heatmap
        3. Selecting the best parameter set
    5. Summary
11. Chapter 6: Trend Pullback Strategies
    1. The market rationale behind a trend pullback strategy
    2. Writing trend pullback components
        1. Existing trend
        2. Pullbacks
        3. Final impulse
    3. Assembling components

        1. Sensitivity analysis
        2. Out-of-sample analysis
    4. Summary
12. Chapter 7: Risk Management
    1. Money management
        1. Price-based exits
        2. Percent-based exits
        3. Volatility-based exits
    2. Position sizing
        1. The P&L equation
        2. Equal dollar risk technique without technical exit levels
        3. Incorporating technical exits in equal dollar risk formulas
    3. Summary
13. Chapter 8: Futures and Forex Algorithmic Trading
    1. Algorithmic trading for futures markets
        1. The basic concepts of futures
        2. How to write algorithms for time breakout strategies
    2. Forex algorithmic trading
        1. The basic concepts of forex
        2. How to write algorithms for the forex market
        3. Breakout versus fake breakout
    3. Summary
14. Chapter 9: The Trading Operational Plan
    1. What is an operational plan?
    2. A fully automated trading plan for futures
        1. How to manage futures symbols for real trading
        2. How to size the four futures portfolio
        3. How to manage live fully automated strategies
    3. A semi-automated trading plan on large stock lists
        1. How to track entries with RadarScreen
        2. How to calculate volatility-based sizes
        3. How to monitor real positions
    4. Summary
15. Chapter 10: EasyLanguage in AI – Bridging Traditional Trading and Advanced Analytics
    1. Python versus EasyLanguage
    2. Volatility Predictor on gold futures by the Gandalf Project

3. Bridging Python and EasyLanguage
4. Embedding AI predictions into EasyLanguage indicators
5. Embedding AI predictions into EasyLanguage strategies
    1. Using the Volatility Predictor model as a filter
    2. Using the Volatility Predictor model for money management
    3. Using the Volatility Predictor model for position sizing
6. Creating a volatility dashboard for multiple assets
7. Using TradeStation to collect predictions from multiple AI models
8. Summary

16. Chapter 11: EasyLanguage for Machine Learning
    1. A definition of machine learning for pattern recognition
    2. The Iris dataset and Fisher’s project
    3. Labeling trading sessions
    4. Project N.1—recognizing an up session
        1. Selecting the target
        2. Selecting the features
        3. Building the pipeline
    5. The confusion matrix
    6. Project N.2—recognizing down sessions
        1. Selecting the target
        2. Selecting features
        3. Final results
    7. Summary
17. Index
    1. Why subscribe?
    2. Packt is searching for authors like you
    3. Share Your Thoughts
    4. Download a free PDF copy of this book

Landmarks

1. Cover
2. Table of Contents
3. Index